Models and algorithms for the Hydro Unit Commitment problem

Alexandre Heintzmann

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Université de Toulouse

Doctorat de l'Université de Toulouse

préparé à Toulouse INP

Modèles et algorithmes pour l'optimisation de la production hydro-électrique

---

Thèse présentée et soutenue, le 5 avril 2024 par

Alexandre HEINTZMANN

École doctorale
SYSTEMES

Spécialité
Informatique

Unité de recherche
LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes

Thèse dirigée par
Christian ARTIGUES, Sandra Ulrich NGUEVEU et Cécile ROTTNER

Composition du jury
Mme Claudia D'AMBROSIO, Présidente, CNRS Ile-de-Franece Gif-sur-Yvette
M. Eduardo MORENO, Rapporteur, Universidad Adolfo Ibañez
M. François CLAUTIAUX, Rapporteur, Université de Bordeaux
Mme Sophie DEMASSEY, Examinatrice, Mines ParisTech
M. Christian ARTIGUES, Directeur de thèse, CNRS Occitanie Ouest
Mme Sandra Ulrich NGUEVEU, Co-directrice de thèse, Toulouse INP
Mme Cécile ROTTNER, Co-directrice de thèse du monde socio-économique, EDF Labs

Membres invités
Mme Pascale BENDOTTI, EDF Labs

Abstract in English

The Hydro Unit Commitment problem (HUC) is a difficult problem playing a major role in the scheduling of daily electricity production at EDF. In this thesis, we study different models and algorithms to solve the special case of the single-plant HUC problem (1-HUC). Studying this case is relevant for the following reasons. On the one hand, there are real world instances of the 1-HUC problem which cannot be solved efficiently by current approaches. On the other hand, it makes it possible to study individually two particular sources of difficulty. One stems from the presence of non-linearities, in particular the power which is a non-convex non-concave function of the flow and the reservoirs' volume. The other is due to the set of hydraulic constraints, specifically the volume minimum and maximum bounds, as well as target volumes for the reservoirs.

In a first part, modeling alternatives for the non-linear 1-HUC, focusing on the power function, are proposed. The aim is to identify a model which can be solved efficiently, with a good trade-off between computational time and precision. The seven proposed modeling alternatives cover a large panel of modeling families. These models are compared on a set of instances with variations on five features that impact the solution. This comparative study enables us to identify three efficient types of models: a model with second order polynomial functions, a model with a piecewise linear function, and a model using a finite set of flows. As the latter model is similar to the current model at EDF, in the following we present algorithms dedicated to it.

In the second part, a polyhedral study is proposed to improve the solving approach of the 1-HUC problem. The idea is to focus on the combinatorial aspects, which means considering the relationship between the bounds on volumes and the discrete set of flows. For this purpose, we introduce a variant of the knapsack problem, with Symmetric weight and Chain Precedences (SCPKP). For the SCPKP, we characterize necessary facet-defining conditions, which are also proven to be sufficient in some cases. A two-phase branch-and-cut algorithm based on these conditions and on the symmetric aspect of the SCPKP is devised. The efficiency of this algorithm is then shown experimentally against state-of-the-art algorithms. The results of this polyhedral study of the SCPKP, as well as the proposed algorithms, are then extended to the 1-HUC problem.

In the third part, an efficient solving technique based on a graph representation of the 1-HUC problem is proposed, in particular to account for the hydraulic constraints. It appears that the 1-HUC problem is a special case of the Shortest Path Problem with Resource Windows (RWSPP). We propose two graph algorithms. The first one is an exact variant of the A* algorithm, using a dual bound dedicated to the 1-HUC problem. In comparison with two state-of-the-art approaches, we show numerically that this algorithm is more efficient for handling a specific case of 1-HUC. The aim of the second algorithm is to take into account a wider set of hydraulic constraints. The idea is based on the concept of bi-objective optimization, for which the second objective corresponds to a relaxation of the volume. The advantage compared to a classical bi-objective optimization is that it is possible to use the minimum and maximum bounds on the volume to reduce the search space and to guide the enumeration of solutions. We show numerically, on a large set of real instances, that this algorithm outperforms three state-of-the-art ap-

proaches. Although this algorithm was designed to solve the 1-HUC problem, it is defined in a generic way for any RWSPP with a single resource.

Résumé en français

Le problème de gestion de production hydro-électrique (HUC) est un problème difficile, qui joue un rôle important dans la gestion de production électrique journalière à EDF. Dans cette thèse, nous étudions différents modèles et algorithmes pour résoudre un cas particulier du problème HUC à une usine (1-HUC). Etudier ce cas particulier présente plusieurs intérêts. D'une part, il existe des instances réelles à une usine mal résolues par les approches actuelles. D'autre part, cela nous permet d'étudier plus spécifiquement deux sources de difficultés séparément. L'une provient de la présence de non-linéarités, notamment la puissance qui est une fonction non-convexe et non-concave du débit d'eau et du volume des réservoirs. L'autre est due à l'ensemble des contraintes hydrauliques, notamment des volumes minimaux et maximaux ainsi que des volumes cibles des réservoirs.

Dans une première partie, différentes alternatives de modélisation des non-linéarités du 1-HUC, plus particulièrement sur la puissance, sont proposées. L'objectif est d'identifier un modèle pouvant être résolu efficacement avec un bon compromis entre temps de calcul et précision. Les sept alternatives proposées couvrent un large spectre de familles de modélisation. Elles sont comparées sur un jeu d'instances présentant des variations sur cinq caractéristiques qui impactent la résolution. Cette étude comparative permet d'identifier trois types de modèles pertinents: un modèle avec des fonctions polynomiales de second ordre, un modèle avec une fonction linéaire par morceaux, et un modèle utilisant un ensemble fini de débits. Ce dernier modèle étant similaire à la modélisation actuelle à EDF, nous proposons dans la suite des algorithmes dédiés à celui-ci.

Dans une deuxième partie, une étude polyédrale est proposée pour améliorer la résolution du problème 1-HUC. L'idée est de focaliser sur le coeur combinatoire, ce qui revient à considérer le lien entre les contraintes sur les volumes et l'ensemble discret des débits. Pour celà, nous définissons une variante du problème du sac-à-dos avec chaines de précédence et poids symétriques (SCPKP). Pour le SCPKP, nous définissons des conditions nécessaires de facettes, qui sont aussi prouvées suffisantes dans certains cas. Un algorithme de branch-and-cut en deux phases s'appuyant sur ces conditions et sur l'aspect symétrique du SCPKP est mis au point. L'efficacité de cet algorithme est ensuite montrée numériquement face à des algorithmes de l'état de l'art. Les résultats de cette analyse polyédrale du SCPKP, ainsi que l'algorithme de résolution proposé sont ensuite étendus au problème 1-HUC.

Dans une troisième partie, une technique de résolution efficace est proposée pour prendre en compte les contraintes hydrauliques en s'appuyant sur la représentation du problème 1-HUC par un graphe. Cela permet de se ramener à un cas particulier du problème de plus court chemin avec fenêtres de ressource (RWSPP). Nous proposons deux algorithmes de graphes. Le premier algorithme est une variante exacte de l'algorithme A*, utilisant une borne duale dédiée au problème 1-HUC. En comparaison avec deux approches de l'état de l'art, nous montrons numériquement que cet algorithme est plus efficace pour traîter un cas spécifique du 1-HUC. L'objectif du second algorithme est de prendre en compte davantage de contraintes hydrauliques. Le principe s'appuie sur le concept d'optimisation bi-objectif pour lequel le second objectif correspond à une relaxation du volume d'eau. L'avantage par rapport à une optimisation bi-objectif classique est qu'il est possible d'utiliser les volumes minimaux et maximaux pour

réduire l'espace de recherche et diriger l'énumération de solutions. Nous montrons numériquement, sur un grand jeu d'instances réelles, que cet algorithme est plus performant que trois approches de l'état de l'art. Même si cet algorithme a été conçu pour résoudre le 1-HUC, nous le définissons de manière générique pour tout RWSPP avec une ressource.

Table of contents

1 Introduction

1.1 Context

Electricity is a commodity for which the world-wide demand grows regularly. Data show that the yearly electricity consumption has increased by a magnitude of four between 1981 and 2019 [1,55]. In [55] this growth in demand appears in many sectors. Large electricity providers hence face the challenge of generating enough electricity to cover this growing demand. Besides, electricity can hardly be stored at large-scale, meaning that most excess of production is lost. Consequently, the aim is to meet exactly the demand. Electricité de France (EDF) is one of these large electricity providers, mainly producing and managing electricity in France. In France, the yearly electricity demand has been stable for the last decade. Nevertheless, meeting the demand remains a challenge as there is no such stability when looking at smaller time scales. **Figure 1.1** shows the monthly electricity demand and production in France between January 2007 and November 2023, with data from [90]. In this figure, it clearly appears that the demand highly varies from a month to another. Similarly, the demand can greatly vary within a day [22], as there can be a factor two between the lowest and the highest demand during the same day. These large variations at different time scales make it difficult for electricity providers to meet the demand. This is particularly true as there is a growing use of renewable energy, some of which can hardly be controlled. For instance solar and wind energy highly depend on the weather. In France, there is twice as much use of renewable energy in 2020 than in 2000 [100]. The use of solar and wind energy has exponentially increased during this twenty-year period. Moreover, for plants using other sources of energy, there are constraints which may not allow the production to follow the demand, especially when the latter increases or decreases quickly. Besides, managing the electricity production at large-scale means managing a large number of plants. For example, at EDF there are more than 600 hydro plants and 56 nuclear reactors. Moreover, there are multiple sources of energy (nuclear, gas, hydraulic, solar, ...) most of them are processed with units, each being a combination of a turbine and a generator. Naturally, each type of unit brings its own set of constraints, which must be taken into account. All of these difficulties lead to an overall excess of production compared to the demand as shown in **Figure 1.1**.

Planning the electricity production in order to meet a forecast demand is commonly known as the Unit Commitment Problem (UCP). This problem is solved over a long-term, a mid-term and a short-term

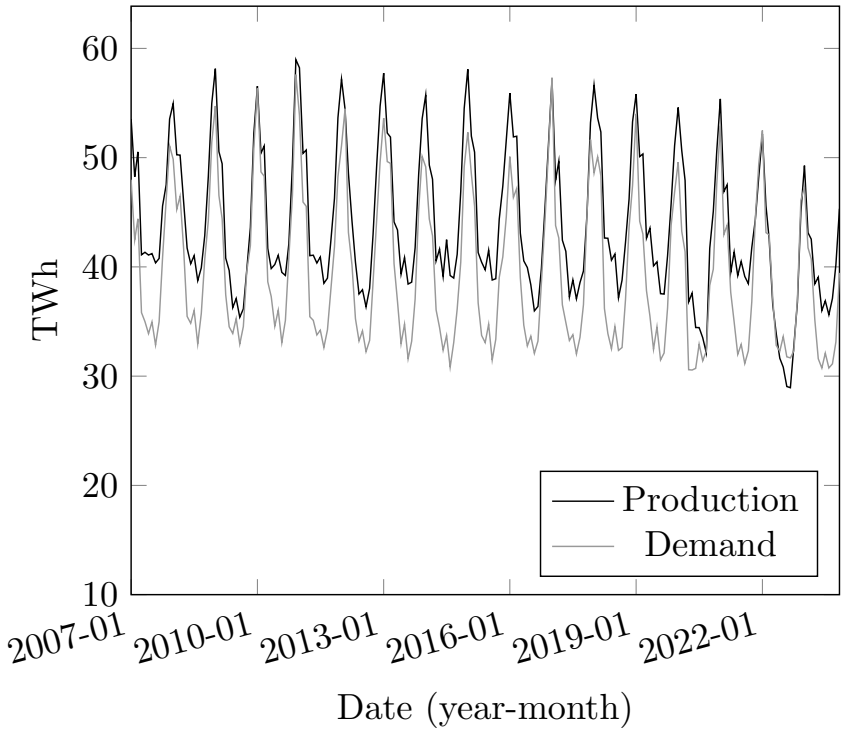


Figure 1.1: Electricity production and demand in France.

horizon. The long-term UCP considers multiple decades, and is related to investments and construction of new plants. The focus is to make strategic decisions in order to anticipate future needs and long-term plans. Such a plan can be for instance to reduce CO_2 emissions or to reduce the energy or resource dependency from other companies or countries. The mid-term UCP provides a general estimation of the production and a cost-effective use of limited resources (fuel, water, ...) over few years. This is done by solving a rough model, where many constraints are omitted and units of a plant are aggregated. The point is to study the big picture, rather than detailing each unit. The mid-term UCP can also be used to schedule operations such as plant maintenance. For the short-term UCP, the aim is to schedule the production of a plant with precise models. Solving efficiently the short-term UCP is mandatory, as it is solved on a daily basis. Due to the number of plants and their variety, the short-term UCP is solved at EDF with a Lagrangian decomposition [87], which creates a sub-problem for each type of production plant. In such decomposition, the demand is satisfied by the master problem. For the sub-problem, the production if guided by prices provided by the master problem.

The Hydro Unit Commitment (HUC) problem is the sub-problem dedicated to hydro valleys and is particularly difficult to solve. One reason is that a valley can be made of multiple cascaded plants, which form a chain of plants linked by reservoirs. This yields coupling constraints between plants, meaning that they cannot be considered independently. A second reason is the intrinsic difficulty of the problem: as presented later, even with a single plant some real-life instances encountered at EDF cannot be solved

to optimality in reasonable times with current methods. Another reason is linked to the numerical errors that can appear when solving real world instances of the HUC problem [91]. This makes the HUC problem harder to solve, but can also lead to suboptimal or infeasible solutions. In particular, infeasible solutions raise severe issues, as they need to be corrected, in order to be used in practice.

At EDF, various ways of solving the HUC problem have been implemented, in order to overcome these difficulties. In [87] a two-phase approach that was used few decades ago at EDF is described. The first phase aims to solve a simpler version of the HUC problem modeled as a Linear Problem, omitting many constraints. The second phase schedules the plants as closely as possible to the solution of the first phase while taking into account the omitted constraints. This second phase sequentially schedules the plants one by one via dynamic programming. The issue of this approach is that optimality is not guaranteed, and scheduling the plants one by one can lead to infeasibility. The current approach at EDF consists in solving a Mixed Integer Linear Program (MILP) modeling a whole valley of the HUC problem [51]. This approach suffers particularly from numerical errors, and can lead to large computational times in some cases. Hence, it may happen that the MILP solving must be stopped before proving optimality or sometimes before finding a feasible solution. Recent work has shown that it is possible to use a Lagrangian decomposition to break down the HUC problem into multiple single plant HUC problems [3] that will be denoted as 1-HUC problems throughout the thesis. In such decomposition, each 1-HUC problem is solved independently by a dynamic programming algorithm [2]. The master problem of the Lagrangian decomposition manages the coupling constraints, i.e., the reservoir constraints located between two plants. This new approach shows promising results. However, the dynamic programming algorithm does not take into account all of the 1-HUC problem's constraints. Hence, the solution provided by this approach may not be feasible in practice.

In this thesis, we propose models and solution approaches for the 1-HUC problem. This problem is relevant since, besides the fact that, as aforementioned, it is possible to decompose a valley with cascaded plants into 1-HUC problems, there also exist real-world valleys restricted to a single plant.

1.2 Definition of the 1-Hydro Unit Commitment problem

The main characteristics of the 1-HUC problem considered in this thesis are as follows. Consider a valley with a single plant, located between an upstream and a downstream reservoir. The plant is composed of K units with a pre-defined start-up order, each unit being the combination of a turbine and a generator. In this thesis, we do not consider the units to be identical. From the hydroelectric production principle, the water from the upstream reservoir flows through the plant to the downstream reservoir. This operates the units of the plant, generating electricity. Some plants can have units operating in reverse, in order to pump the water from the downstream reservoir to the upstream reservoir, consuming electricity. The units can have different characteristics, such as the efficiency with respect to the flow, and have a prescribed start-up order. **Figure 1.2** shows a diagram of a hydroelectric plant.

The time horizon, being 48 hours at EDF, is discretized in $T = 96$ time periods, each of duration

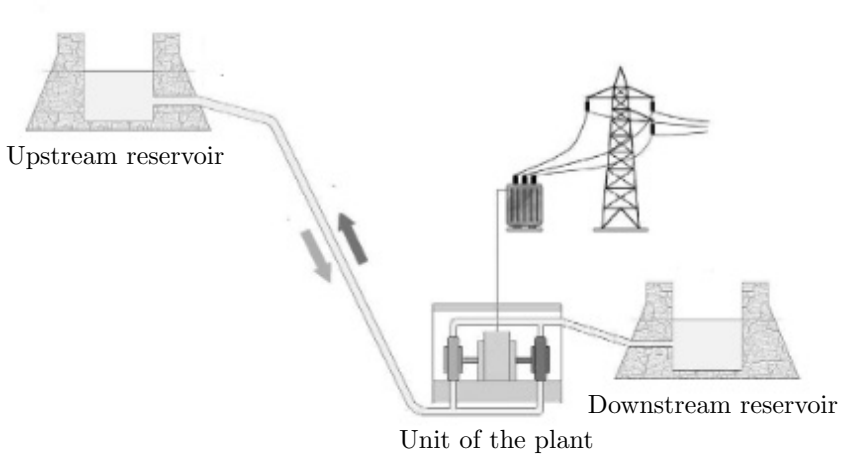


Figure 1.2: Diagram of the 1-HUC problem

Δ equal to 30 minutes. The water flow through the plant at each time period t is the main decision variable of the problem. The constraints of the 1-HUC problem considered in the thesis are as follows. The flow lies in interval $[\underline{D};\overline{D}]$. We consider the general case where a plant cannot turbine and pump simultaneously. The ramping constraints limits the water flow variation from time period t to $t+1$. The min-up constraints indicate that if the water flow increases, it cannot decrease during a fixed number of time periods. Similarly, the min-down constraints indicate that if the water flow decreases, it cannot increase during a fixed number of time periods. Each reservoir $n \in \{1,2\}$ has an initial volume V_0^n as well as a time-dependent maximum capacity $\overline{V}_t^n$ and time-dependent minimum capacity $\underline{V}_t^n$. These bounds form resource windows, the water in the reservoirs being the resource. Note that resource windows differ from time windows [56]. Indeed, in the case of time windows, waiting time is allowed. Hence, time windows are less restrictive than resource windows. These time-dependent maximum and minimum capacities make it possible to set target volumes for specific time periods, when $\overline{V}_t^n = \underline{V}_t^n$. Introducing target volumes accounts for water management policies. At EDF, these target volumes are defined on a year ahead basis, resulting from cost-efficient use of the resources obtained with the mid-term UCP. In such a case, target volumes may be present only for the last time period of each day, meaning that there can be up to two target volumes for instances covering 48 hours.

At each time period, the reservoir n has an additional intake of water A_t^n, which can be positive (rain, melting snow) or negative (use of water for surrounding agriculture). We consider the energy to have a time-dependent unitary value Λ_t. At the end of time period T, the water in reservoir n has an expected unitary value Φ^n. High Φ^n values will lead to preserve more water and produce less energy, and the other way around for low Φ^n. At EDF, the HUC problem is considered as a revenue-maximizing price-taker scheduler problem. In this case, many characteristics of the instance are fixed data. the power prices, the water expected value and the reservoir external inflows and reservoirs capacities are considered as fixed data. Naturally, the number of units, the rate of the ramping constraints and the duration of the min-up/down constraints depend on the plant. Similarly, the maximal and minimal volumes (without

target volumes) and the initial volume depend on the reservoir. The target volumes, water expected value and external inflows are provided by the mid-term UCP. The power prices come from the master problem of the Lagrangian decomposition.

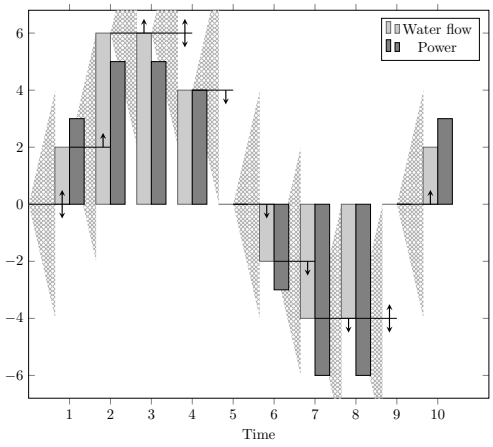
In the following example, we show a valid solution for a simplified instance of the 1-HUC problem.

Example 1 (*Simplified instance of the 1-HUC problem*)

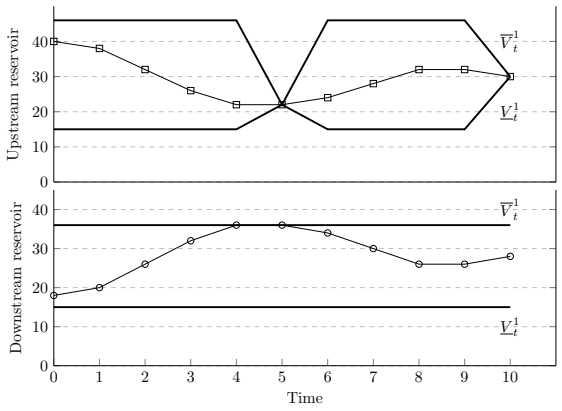
Consider a simplified instance of the 1-HUC, with $T = 10$. *The initial volumes are* $V_0^1 =$, $V_0^2 = 18$. *For the upstream reservoir, the maximum and minimum capacities are* $\overline{V}_t^1 = 46$ *and* $\underline{V}_t^1 = 15$ *for all time periods* t *besides 5 and 10 where targets volumes are* $\overline{V}_5^1 = \underline{V}_5^1 = 22$ *and* $\overline{V}_{10}^1 = \underline{V}_{10}^1 = 30$. *For the downstream reservoir, the maximum and minimum capacities are* $\overline{V}_t^2 = 36$ *and* $\underline{V}_t^2 = 15$ *for all time periods. The flow lies in interval* $[\underline{D} = -4; \overline{D} = 6]$. *Ramping constraints are such that the flow can increase or decrease by at most 4 per time period. Min-up/down constraints are such that if the flow increases (resp. decrease), it cannot decrease (resp. increase) during a total 2 time periods. For this example, there are no external intakes. The power unitary values are* $[10.5, 11.8, 13.2, 10.2, 8.6, 9.4, 9.5, 7.4, 8.9, 9.7]$. *The unitary value of the reservoirs are* $\Phi^1 = 11.3$, $\Phi^2 = 5.6$.

Figure 1.3 *shows an example of a solution. The bargraph in* ***Figure 1.3a*** *shows the power in dark grey and the flow in light grey at each time period. Moreover, for each time period, the crosshatched cones represent the ramping constraints and the arrows point up (resp.down) if the min-up (resp. min-down) constraints is satisfied.* ***Figure 1.3b*** *shows the evolution of the volume as well as the upper and lower bounds at each time period relative to the upstream and downstream reservoirs.*

The value of the power produced by this solution is 96.8. The value of the volume in the upstream reservoir is 339 and in the downstream reservoir is 156.8. The total value of this solution is 592.6.



(a) Power and water flow at each time period

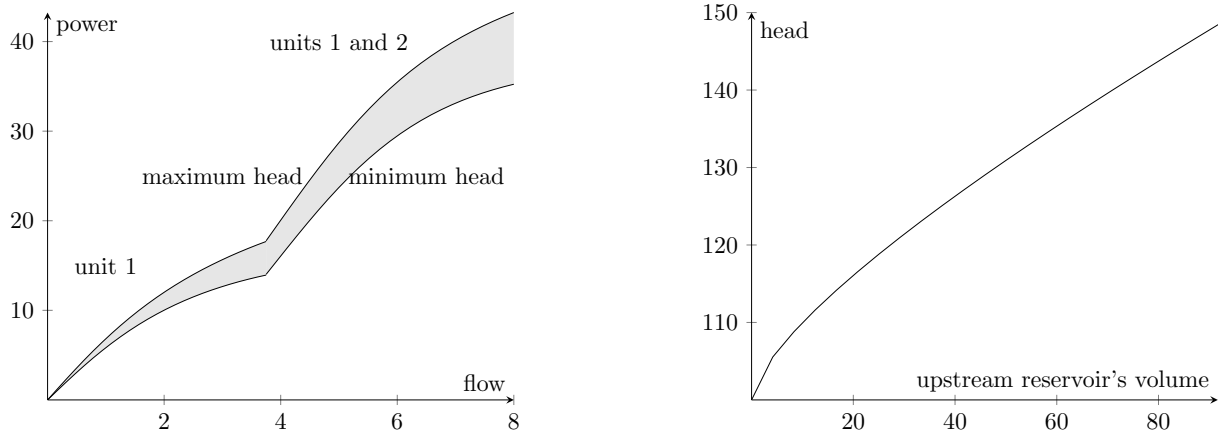


(b) Volume of the upstream and downstream reservoirs at each time period

Figure 1.3: Power, flow and volumes at each time period

Some characteristics of the 1-HUC problem have not been presented yet: the ones that prescribe how

the water flow is transformed into power. As shown in **Figure 1.3a**, the power is not linear with respect to the flow, meaning that the 1-HUC problem is a non-linear problem. Indeed, the power production is in fact a non-linear non-convex non-concave function of the water flow and the water head. The water head is the vertical distance between the water level of the upstream reservoir and the downstream reservoir. This is because the turbine efficiency of each unit is concave, as presented in [5]. This means that for a given unit, the higher the flow, the lower is the efficiency with respect to the flow. Besides, the plant has K units which start-up in a prescribed order. If we wanted to represent the power with a single function of the water flow, it results in a non-convex non-concave function. Modeling choices will be discussed in **Chapter 2**. **Figure 1.4a** gives the shape of the power function in the case of two units, for the maximum and the minimum head. As the reservoirs are larger at the top, and tighter at the bottom, the head is a non-linear function of the volumes. **Figure 1.4b** gives an example of the head with respect to the upstream reservoir. A similar curve exists for the downstream reservoir, but for which the head is higher when the volume in the downstream reservoir is low. A modeling challenge is to define a mathematical function giving the produced power depending on the water flow that provides a trade-off between precision and computational tractability in solution approaches.



(a) Shape of the power function with respect to the flow with $K = 2$ for the minimum and maximum head

(b) Shape of the head with respect to the upstream reservoir's volume

Figure 1.4: Shape of the power function and the head function

Solving the 1-HUC problem consists in scheduling the power production of the plant, such that the value of the valley is maximized while satisfying the capacities at each time period, the ramping constraints and the min-up/down constraints.

In the following, we present the three main parts of this thesis, corresponding to three different modeling and solution approaches for the 1-HUC problem: non-linear modelings, polyhedral studies and graph algorithms. As these approaches involve quite different research areas, we chose to present a literature review in each chapter rather than a unique review for the whole thesis. Besides, different variants of the 1-HUC are studied, as well as different models for each of them. **Figure 1.5** depicts each 1-HUC variant introduced, and the models considered for each of them in the thesis.

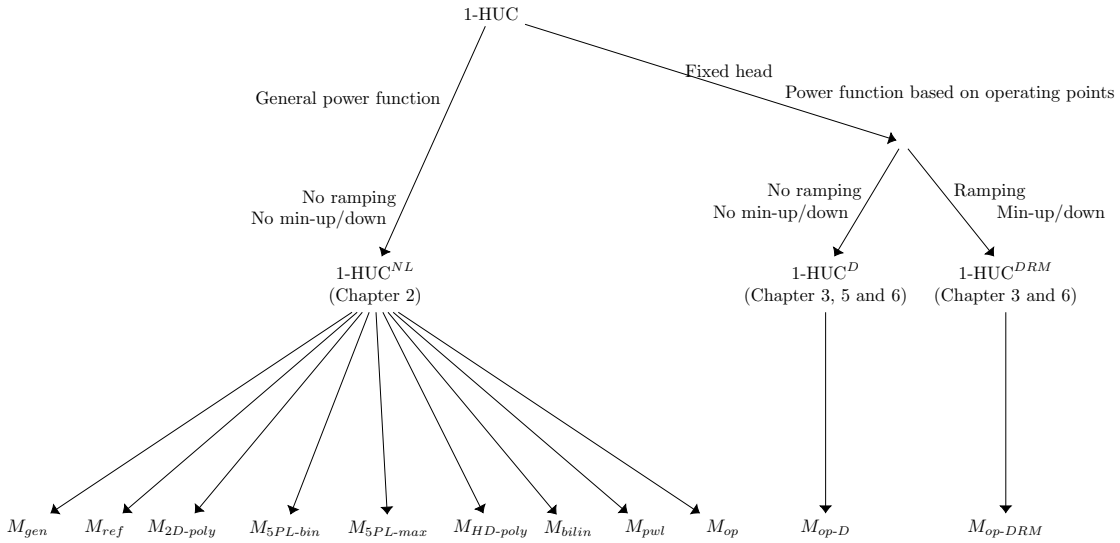


Figure 1.5: Variants of the 1-HUC problem and models considered in the thesis

1.3 Part I: Modeling choice

In **Chapter** 2, we compare various modeling alternatives for the 1-HUC problem, focusing mainly on the power function, which is non-convex non-concave, and can be difficult to handle efficiently. Besides, there are many cases without an analytical representation of the power function [34]. Hence, providing and studying tractable power functions could lead to a quicker solution process, but also improve the quality of the solutions obtained. This is why studying modeling alternatives for the power function of the HUC problem has raised interest in the recent literature [96,34]. For the purpose of this study, we consider a simplified version of the 1-HUC problem, namely the 1-HUC^{NL} problem. This choice has been made to study specifically the power function, even though adding additional contraints could help MINLP solvers. Indeed, the 1-HUC^{NL} problem does not take into account ramping constraints, min-up/down constraints and pumps. Moreover, the head for this simplified problem is the height difference between the surface of the water in the upstream reservoir and the plant, rather than between the surface of the upstream and the downstream reservoir. Since the 1-HUC^{NL} problem features a single plant, there is no downstream plant to turbine the incoming water in the downstream reservoir, which could amplify the head effect. For a more comprehensive study, considering a downstream plant would be needed, which is beyond the scope of this comparison.

This simplified problem is represented by model M_{gen} defined in **Section 2.1**. Considering such a simplification makes it possible to clearly see the efficiency of the modeling alternatives when studying the power function. The aim is to detect the best modeling alternatives, in terms of computational time, feasibility and approximation.

In a systematic study that goes beyond the comparisons already made in the literature, we compare seven alternatives. This includes common models from the literature as well as new models that did not

exist in the literature of the HUC problem. As we compare a larger set of models, we also cover a wider family of models: with or without linear functions, with a unique or multiple functions, with or without binary variables. We also study the impact of five characteristics of the 1-HUC^{NL} problem instances on the performance of the compared models. Comparisons of this study are also carried out for the special case of the non-linear 1-HUC problem where the head is fixed. This is a common simplification, as there are cases where the reservoirs are very large, and the volume variation through a day is negligible. In this case, the head is a constant and the power becomes a non-linear function of the water flow only.

The comparisons are focused on modeling alternatives, not on solution approaches. All the proposed models are solved by off-the-shelf solvers. For each modeling alternative, only a baseline model is considered, and we do not include state-of-the-art methods that could improve these baseline models. Due to the large variety of non-linear functions, a large number of tools exist to manage non-linear functions. Each solver implements its own set of tools, meaning that a non-linear function featured in a model can be more efficiently taken into account by one solver than another. Hence, we consider five different non-linear solvers, and we identify for each model the most adequate one.

Based on this comparison, we identify four models that stand-out in the general case, and three models in the fixed-head case. Indeed, these models give the best trade-offs between metrics such as the computational time, the precision and feasibility, making them reliable without too large computational times. These results are consistent with the literature of the HUC problem, as these modeling alternatives that stand out frequently encountered in the literature.

In **Chapter 3**, we select one of the modeling alternatives for the remainder of the thesis, where we focus on studying hydraulic constraints, such as the volume bounds, the ramping and the min-up/down constraints. Such constraints are not impacted by the head, hence we chose one of the efficient model in the fixed-head case. More precisely, the three models that stand out are the following: $M_{2D\text{-}poly}$ involving polynomial functions of degree two, model M_{PWL} using a piecewise linear function, and model M_{op} based on a finite set of flows. Model $M_{2D\text{-}poly}$ provides low approximation error, with reasonable computational times as polynomial functions are common non-linear functions. Model M_{PWL} is linear, making it possible to use efficient MILP approaches. Model M_{op} can reach very low approximation error, and is linear in the fixed-head case. As M_{op} yields the lowest approximation error out of the three, reducing the time required to solve it would produce an effective approach. This motivates us for generalizing this model and focusing on it in the remainder of the thesis.

When the head is fixed, each flow is associated with a specific power produced. These pairs flow/power are commonly defined as operating points and are defined in a cumulative manner at EDF. We introduce the 1-HUC^{DRM} problem, being a discretized variant of the fixed-head 1-HUC through operating points, with ramping and min-up/down constraints. We present a model for the 1-HUC^{DRM} problem, denoted $M_{op\text{-}DRM}$. We then reformulate the maximum and minimum bounds on the volumes for $M_{op\text{-}DRM}$. Doing so makes it possible to identify that the bounds on the volumes can be represented in two fashions. The first one is as a resource window on the cumulated flow, i.e., the sum of the flow since time period 1. The second one is as nested knapsack inequalities and covering inequalities, the former providing an upper bound and the latter providing a lower bound on the cumulated flow. Finally, we present a

bound-tightening procedure for these constraints. This procedure is particularly interesting as it can be done in polynomial time and can improve the efficiency of many approaches. For the remainder of the thesis, we also introduce the 1-HUC^{D} problem, which is the special case of the 1-HUC^{DRM} problem without ramping nor min-up/down constraints. The corresponding model $M_{op\text{-}D}$ can also be enhanced in the same way as model $M_{op\text{-}DRM}$.

Note that nested knapsack constraints is different from the nested knapsack problems [57]. In our case, the constraints are nested as they apply on different set of variables, which are all subset of each other. In the case of the nested knapsack problems, and one must take into account the capacity and the value of multiple subsets. Besides, in the latter case, the subsets are disjoint.

1.4 Part II: Polyhedral studies

Conducting a polyhedral study of the 1-HUC^{DRM} problem is motivated by the fact that the current approach at EDF to solve the HUC problem is based on a MILP model such as $M_{op\text{-}DRM}$. The objective is then to use the results of this polyhedral study to improve the current solution approach. The idea is to focus on the combinatorial aspects, which means considering the relationship between the upperbound on cumulated flow and the operating points. For this purpose, we define a variant of the knapsack problem, with Symmetric weight and Chain Precedences (SCPKP).

In **Chapter 4**, we conduct a polyhedral study regarding the SCPKP. The SCPKP is a variant of the Knapsack Problem (KP) [7] that has not been studied yet in the literature. In order to define the SCPKP, we define the Chain Precedence Knapsack problem (CPKP) as follows. Consider I groups of J elements, where I and J are positive integers. Let item (i, j) denote element j of group i. Item (i, j) has a weight $W_{ij} \in \mathbb{R}_{\geq 0}$ and a value $V_{ij} \in \mathbb{R}$. Within each group, order constraints are such that any item (i, j) can be selected only if item $(i, j-1)$ is selected, thus inducing chain precedence constraints. Let C be the maximum knapsack capacity. Solving the CPKP consists in maximizing the total value of the selected items, while the chain precedence constraints are satisfied, and the total weight of the selected items is less than or equal to capacity C. The SCPKP is a CPKP where item (i, j) has a weight $W_{ij} = W_j$, i.e., the weight of item (i, j) does not depend on the group index i. It means that items (i, j) and (i', j) have the same weight, thus the knapsack is symmetrically weighted with respect to the groups. Clearly, as the chain precedence constraints are a special case of precedence constraints, the SCPKP is a special case of the Precedence Knapsack Problem (PKP) [17]. **Figure 1.6** depicts a graphical representation of the SCPKP, where a square represents an item, and arrows stand for the chain precedence constraints.

The interest of studying the SCPKP is that it is the combinatorial core of the 1-HUC^{DRM} problem. Indeed, the capacity constraint of the SCPKP acts like one of the nested knapsack constraint or resource constraint of the 1-HUC^{DRM} problem. Besides, each group of items of the SCPKP reflect the behaviour of the operating points for a time period of the 1-HUC^{DRM} problem. Moreover, one particular aspect of the 1-HUC^{DRM} problem that is captured by the SCPKP is the symmetry of feasibility for the solutions. In the case of the SCPKP, symmetry appear due to symmetric weights. Similarly, in the case of the 1-HUC^{DRM}

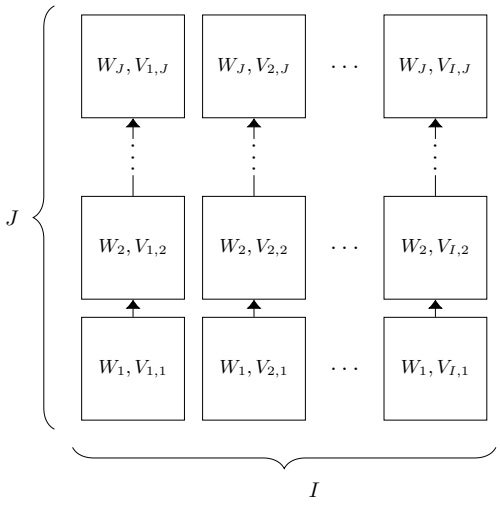


Figure 1.6: Graphical representation of the SCPKP

problem, symmetry appear as the operating points are the same for each time period. Hence, the amount of flow used for an operating point is identical for all time periods. It is important to take this symmetry into account as it appears in the facet-defining inequalities for both the 1-HUC^{DRM} problem and the SCPKP.

In order to handle inequalities without considering all the symmetries explicitly, we introduce patterns representing all symmetries of an inequality. During the polyhedral study of the SCPKP, we define three necessary facet-defining conditions for patterns. We also identify a family of patterns for which these conditions are sufficient, hence ensuring that all associated inequalities are facet-defining. In order to exploit these patterns, we present a two-phase Branch & Cut (B&C) algorithm. The first (preprocessing) phase aims to generate patterns satisfying the three necessary facet-defining conditions. A second phase consists of solving the SCPKP with a B&C algorithm, using pre-computed patterns to generate cuts. This two-phase approach, using pre-calculated patterns in the B&C, is motivated by the following reasons. Patterns satisfying the three facet-defining conditions can be generated independently of the B&C algorithm. Then, we conjectured that the separation problem is NP-hard, but can be solved in polynomial time for a given pattern. Finally, we can manage the set of pre-calculated patterns efficiently by identifying during the B&C the ones providing the best cuts for the current instance.

The proposed two-phase B&C is compared experimentally against two other B&C variants, namely default CPLEX and a classical B&C scheme featuring state-of-the-art separation of PKP cover inequalities, adapted to the SCPKP. The latter is relevant as the SCPKP is a special case of the PKP. Hence, valid inequalities for the PKP are also valid for the SCPKP. Results show that the two-phase B&C algorithm is the most efficient out of the three B&C variants. Besides, we also compare various combinations of these variants. The results show that the most efficient variants always feature our two-phase B&C algorithm.

In **Chapter 5** we extend the results of the SCPKP to the 1-HUC^{D} problem modeled with $M_{op\text{-}D}$. The reason why we consider the 1-HUC^{D} problem rather than the 1-HUC^{DRM} is because the SCPKP does

not include ramping and min-up/down constraints. As such, the relationship between the SCPKP and the 1-HUC^D problem appears more clearly.

In order to extend the polyhedral study of the SCPKP, we first define the Inverted SCPKP (ISCPKP), which is an SCPKP but with a covering inequality instead of a knapsack inequality. We demonstrate that the ISCPKP can be modeled as an SCPKP, hence the polyhedral study of the latter can be translated to the former. We also show that the reservoir constraints of the 1-HUC^D problem correspond exactly to T SCPKPs and ISCPKPs. However, experimental results show that the facet-defining inequalities of the 1-HUC^D problem are not necessarily linked to these T (I)SCPKPs. On one hand, there are (I)SCPKPs for which facet-defining inequalities are not facet-defining for the 1-HUC^D problem. On the other hand, there are other (I)SCPKPs that share facet-defining inequalities with the 1-HUC^D problem.

Based on these observations, we propose two pre-processing procedures. The first one identifies the relevant (I)SCPKPs among the ones defining the 1-HUC^D problem, i.e., the ones sharing facet-defining inequalities with the 1-HUC^D problem. We prove that this procedure runs in polynomial time. The second one infers new (I)SCPKPs that share facet-defining inequalities with the 1-HUC^D problem. We demonstrate that there are at most a polynomial number of such (I)SCPKP, and they can all be inferred from the (I)SCPKP retained in the first procedure. Once all these (I)SCPKPs are identified, one can adapt the two-phase B&C scheme, initially designed for the SCPKP, to the 1-HUC^D problem.

1.5 Part III: Graph algorithms

In **Chapter 6** we describe graph algorithms to solve the 1-HUC^{DRM} problem modeled with $M_{op\text{-}DRM}$. Indeed, the 1-HUC^{DRM} problem has a specific structure, allowing for representing it with graphs. As stated in **Section 3.2**, the volume upper and lower bounds form resource windows. Hence, the 1-HUC^{DRM} problem can be considered as a Shortest Path Problem with Resource Windows (RWSPP), defined as follows. Let $G = (V, A)$ be a graph, with V the set of vertices and A the set of arcs. We denote s and t respectively the source vertex and the target vertex. Each arc $a \in A$ has a value $V(a) \in \mathbb{R}$ and a resource amount $R(a) \in \mathbb{R}_{\geq 0}$. Each vertex $v \in V$ has a resource window $[\underline{R}(v); \overline{R}(v)]$. For a path C, we denote $V(C) = \sum_{a \in C} V(a)$ the value of the path, and $R(C) = \sum_{a \in C} R(C)$ the amount of resource used. A path C from s to v is locally feasible if $R(C) \in [\underline{R}(v); \overline{R}(v)]$. A path C is feasible if all sub-paths C' of C starting from s are locally feasible. The aim is to find a feasible path C from s to p minimizing $V(C)$. The RWSPP is a generalization of the Resource Constrained Shortest Path Problem (RCSPP), for which only upper bound $\overline{R}(v)$ exists for a vertex $v \in V$. Problems modeled as RWSPP can be challenging, as classical dominance rules of the RCSPP do not apply [2], and there are only a few algorithms dedicated to solve the RWSPP [102].

Note that the RWSPP is different from the shortest path with time windows [56]. In the case of time windows, waiting time is allowed, meaning that the time windows only apply when exiting a state. Hence, time windows are less restrictive than resource windows.

We propose two different graph representations of the 1-HUC^{DRM} problem an an RWSPP. The first

one is a cumulated flow-expanded graph with vertices representing the time period, the operating point and the volume of water used. The second one is a compact graph which considers vertices representing the time period and the operating point. The former makes it possible to take into account the resource windows directly by discarding vertices, but yields a pseudo-polynomial number of vertices, as it depends on the bounds on volumes. The latter cannot represent the resource windows, but has a polynomial number of vertices.

Note that the 1-HUC^{DRM} problem aims at maximizing the value, hence it can be defined as a Longest Path Problem with Resource Windows (RWLPP). However, the graph representations of the 1-HUC^{DRM} problem proposed in this section are acyclic, in which case shortest and longest path problems are equivalent.

Then, we present the Hydro A* (HA*) algorithm on the cumulated flow-expanded graph. This algorithm is an exact variant of the A* algorithm [50], initially designed to accelerate the shortest path algorithms. In order for A* to be exact, it is mandatory to define a dual bound. This is because any feasible solution provides a lower bound (for a maximization problem) on the value of the optimal solution that can be used to discard all partial paths for which the dual bound does not exceed the lower bound. In the case where the bound is not a dual bound, then one needs to enumerate all solutions to guarantee optimality. As it is impractical to enumerate all solutions for most problems, when no dual bound exists, A* algorithm cannot guarantee optimality. For algorithm HA*, we introduce a dual bound specific to the 1-HUC^{D} problem, as it does not take into account the ramping and min-up/down constraints. As the 1-HUC^{D} is less constrained than the 1-HUC^{DRM} problem, a dual bound for the former is also a dual bound for the latter. More precisely, this dual bound computes an improved linear relaxation of model $M_{op\text{-}D}$ corresponding to the 1-HUC^{D} problem. We compare HA* to a classical RCSPP algorithm extended to the RWSPP, as well as to the current approach at EDF, consisting in solving model $M_{op\text{-}DRM}$ with CPLEX, on real size instances. The results show that the proposed algorithm is on average the most efficient for instances of the 1-HUC^{D} problem. However, in general cases of the 1-HUC^{DRM}, the performances of HA* decrease. This is mainly due to the dual bound which omits the ramping and min-up/down constraints.

We then propose a second algorithm, the Bi-Objective relaxation of the Resource Windows (BORWin) algorithm. This algorithm is designed for the compact graph. Unlike HA*, no feature of BORWin is specific to the 1-HUC^{DRM} problem, hence it can be used to solve any RWSPP with a single resource. The main idea of BORWin is to relax the resource windows and to consider the resource as a second objective. We then use similar principles as in the two-phase algorithm for bi-objective problems [103], namely finding restricted search spaces and enumerate solutions in these search spaces. In the case of the RWSPP, there is only a single optimal solution. Hence, during the first phase of BORWin, we do not need to define multiple search spaces, but only the one containing an optimal solution. Moreover, we can also use the resource windows to tighten this search space. The second phase of BORWin enumerates solutions of the relaxed problem inside the obtained restricted search space. To do so, we use a labeling algorithm inspired by an approach to find the K best solutions [62]. The main idea is to generate solutions of the relaxed problem, which are progressively corrected into feasible solutions. For this

second phase, we also generalize the classical labeling extension in order to generate more solutions at once. This generalization helps to build feasible solutions quicker.

Algorithm BORWin is compared experimentally to three alternatives, namely HA*, a classical RCSPP algorithm, and solving model $M_{op\text{-}DRM}$ with a MILP solver, which corresponds to the current practice at EDF. This comparison is made on a wide range of EDF instances for the 1-HUC^{DRM} problem. Numerical results show that the two-phase algorithm largely outperforms HA* and the RCSPP algorithm. Besides, we also note that BORWin is more effective than solving $M_{op\text{-}DRM}$. This is specially true as solving the MILP model introduces numerical errors, which yields infeasible solutions or suboptimal solutions.

In **Chapter 7**, we draw concluding remarks and perspectives to improve and extend the work carried out in this thesis.

1.6 Reading guide for this thesis

The reader interested in the modeling comparison simply needs to read **Chapter 2**. The reader interested in the discretized model selected and its enhancements can jump to **Chapter 3**. The reader interested in the polyhedral study of the Symmetric-weight Chain Precedence Problem can jump to **Chapter 4**. For the reader interested in the polyhedral study of the 1-HUC^{DRM} problem in **Chapter 5**, we recommend reading **Section 3.3** and **4** beforehand. In the former is defined the problem considered and the associated model $M_{op\text{-}D}$ which are used in **Chapter 5**. In the latter are defined the patterns and various concepts related to them as well as the two-phase B&C algorithm, which are extended in **Chapter 5**. For the reader interested in the graph algorithms, in **Chapter 6**, we recommend reading **Section 3.3**, where the problem and the associated model $M_{op\text{-}DRM}$ is defined.

Part I

Modeling choice

2 Modeling comparison for a simplified non-linear 1-Hydro Unit Commitment problem

Table of contents

Real-world problems often involve non-linearities. These non-linearities are not always analytically defined, or can be too complicated to be efficiently handled by a solver. In such a case, one must define a tractable function, amongst multiple modeling alternatives. As mentioned in the previous chapter, the 1-HUC problem falls into such case, where non-linear functions do not always have an analytic representation.

In this chapter, we study modeling alternatives for a simplified version of the non-linear 1-HUC problem, focusing on the non-linear power function. In **Section 2.1** we define the problem considered. In **Section 2.2** we review the power functions defined in the literature of the 1-HUC. In **Section 2.3** we present different modeling alternatives for the 1-HUC problem considered. In **Section 2.4** presents an extensive computational evaluation of the modeling alternatives proposed. In **Section 2.5** we draw concluding remarks.

2.1 The simplified non-linear 1-Hydro Unit Commitment problem and its characteristics

We consider a simplified non-linear 1-HUC problem, namely the 1-HUC^{NL} problem, defined as follows. Consider a valley with a single plant, located between an upstream and a downstream reservoir. The plant is composed of K units. The time is discretized into T time periods, each of duration Δ. At each time period t an amount d_t flows from the upstream to the downstream reservoir, operating the units of the plant. This generates a power p_t, which is non-linear with respect to the water flow d_t and the head h_t. The head is the height difference between the surface of the water in the upstream reservoir and the plant, and is non-linear with respect to the volume of the upstream reservoir v_t^1. At each time period t, the volume of the upstream reservoir v_t^1 (resp. downstream reservoir v_t^2) must lie within the volume bounds $[\underline{V}_t^1; \overline{V}_t^1]$ (resp. $[\underline{V}_t^2; \overline{V}_t^2]$). Moreover, at each time period t and for each reservoir n, there is an external intake of water in the reservoirs, $A_t^n \in \mathbb{R}$. We consider the case without pumps, meaning that $d_t \in [\underline{D}, \overline{D}]$ with $\underline{D} \geq 0$. The power produced at time period t has unitary value Λ_t and the water of reservoir $n \in \{1, 2\}$ has unitary value Φ^t at time period T. Solving the 1-HUC^{NL} problem is to schedule the flow usage of the plant such that the value of the valley is maximized.

A generic model M_{gen} is given as follows:

$$\max \sum_{t=1}^{T} \Delta \cdot \Lambda_t \cdot p_t + \sum_{n=1}^{2} \Phi^n \cdot v_T^n \tag{2.1.1}$$

$$\text{s.t. } v_t^1 = V_0^1 + \sum_{t'=1}^{t} \left(A_{t'}^1 - d_{t'} \Delta \right) \qquad \forall t \leq T \tag{2.1.2}$$

$$v_t^2 = V_0^2 + \sum_{t'=1}^{t} \left(A_{t'}^2 + d_{t'} \Delta \right) \qquad \forall t \leq T \tag{2.1.3}$$

$$h_t = \boldsymbol{f}(v_t^1) \qquad \forall t \leq T \tag{2.1.4}$$

$$p_t = \boldsymbol{F}(d_t, h_t) \qquad \forall t \leq T \tag{2.1.5}$$

$$\underline{V}_t^n \leq v_t^n \leq \overline{V}_t^n \qquad \forall t \leq T, \forall n \in \{1, 2\} \tag{2.1.6}$$

$$\underline{D} \leq d_t \leq \overline{D} \qquad \forall t \leq T \tag{2.1.7}$$

Constraints (2.1.2) and (2.1.3) are volume conservation constraints. Note that it is also possible to define these constraints with consecutive time periods, which yields the same relaxation and similar computational times. Constraints (2.1.4) express the water head h_t, using the concave function $\boldsymbol{f}$ of the volume v_t^1. Constraints (2.1.5) define the power p_t, using the non-convex non-concave non-linear function $\boldsymbol{F}$ of the water flow d_t and the head h_t. Constraints (2.1.6) and (2.1.7) set upper and lower bounds for variables. The criterion to maximize is the profit, which is a linear expression described by (2.1.1). Note that model M_{gen} features equations that could be replaced by inequalities, which makes it possible to use a broader set of resolution tools. However, preliminary results have not shown any improvement, in terms of computational times, approximation error or feasiblity for any of the proposed

models with inequalities rather than equations. Hence, in the following, we define the models with equations.

Model M_{gen} acts as a framework for the different models proposed in this chapter, hence M_{gen} is as generic as possible, functions $\boldsymbol{f}$ and $\boldsymbol{F}$ being not specified. However, we can still specify some characteristics of M_{gen}. The water flow d_t is non-negative because we consider the plant to feature turbines only. Clearly, the volume v_t^n is also non-negative. By definition of the head and the power, both functions $\boldsymbol{f}$ and $\boldsymbol{F}$ are non-decreasing and non-negative. This means that variables h_t and p_t are both non-negative.

Based on data at our disposal, we describe function $\boldsymbol{f}$ and $\boldsymbol{F}$ as follows. The function $\boldsymbol{f}$ is a concave function, as the top of a reservoir is wider than the bottom. The function $\boldsymbol{F}$ is neither concave nor convex. This is because it accounts for the turbines efficiencies, for which we consider a standard non-linearity, that is concave for each unit [5]. Moreover, recall that the units have a prescribed start-up order. Considering multiple units adds non-concavity to the resulting function. More precisely, for each unit the power function is almost linear with respect to the flow when the unit starts, then it incurves more and more until the next unit starts. When another unit starts, we notice a break in the function shape, i.e., a non-differentiable point. An example of the power function and the head function is depicted in **Figure 1.4**. We identify four main characteristics of $\boldsymbol{F}$, described in **Table 2.1**.

Table 2.1: Characteristics of the power function

C1	non-convex and non-concave
C2	locally linear when a unit starts
C3	concave for each unit with respect to the water flow
C4	non-differentiable points when starting up a new unit

A standard simplification of the 1-HUC^{NL} problem is to assume a fixed-head $h_t = H$ where H is a parameter. In the following, we denote such case the *fixed-head 1-HUC^{NL} problem*. This simplification is practically relevant for some instances of the 1-HUC^{NL} problem where head variations are small enough for the impact on the turbines efficiency to be insignificant. For the fixed-head 1-HUC^{NL} problem variant also considered in this chapter, equality (2.1.4) and (2.1.5) from M_{gen} are replaced by:

$$p_t = \boldsymbol{F}(d_t, H) \qquad \forall t \leq T \tag{2.1.8}$$

In such a case, function $\boldsymbol{F}$ becomes a one-dimensional function, but remains non-convex non-concave and all the characteristics of **Table 2.1** hold. Note that even if the head is constant, we still consider variables v_t^1 and v_t^2 in the model, to ensure that reservoir capacities are respected.

2.2 Literature review of the Hydro Unit Commitment power function

When a non-linear function is not analytically defined or tractable, there are multiple possible modeling choices for its approximation. **Figure 2.1a** shows an example of a non-linear function and **Figures**

2.1b to **2.1e** depict four classical approximations. Each of them has benefits and drawbacks which impact its results, both in terms of computational time and quality. In this literature review, we aim at identifying the various modeling alternatives for the non-linear functions considered in the 1-HUC as illustrated in **Figure 2.1**. We focus on the power function as a non-linear function of the water head h_t and the water flow d_t at each time period t: $p_t = \boldsymbol{F}(d_t, h_t)$. The head is a non-linear function of the volume at each time period: $h_t = \boldsymbol{f}(v_t)$. We also review some modeling comparisons proposed in the literature. However, the solutions approaches are not the focus of this review, as in this chapter we consider off-the-shelf solvers to solve the modeling alternatives.

As mentioned in [34], there are cases of the HUC where no perfect analytic representation of the hydroelectric power function is known. Nevertheless, some functions have been described in the literature as a baseline to measure the approximation error of the various models proposed in the literature. A generic simplified hydroelectric power function [46] is:

$$\boldsymbol{F}(d_t, h_t) = \rho \cdot G \cdot h_t \cdot d_t \tag{2.2.1}$$

which defines the power function as a product of ρ the density of water, G the gravitational constant, d_t the water flow and h_t the head. Note that this generic bilinear function is non-convex and non-concave [84].

In terms of modeling, some (MI)NLP featuring (2.2.1) or a similar equation are described in the literature. Indeed, a bilinear function is a common non-linear function, well handled by non-linear solvers even for large-scale instances. In [70] the HUC considered has multiple cascaded plants, and is modeled as an NLP, with the power defined as a bilinear function depending on the water flow and the head. In [67], an algorithm called spatial Hydro Branch and Bound (sHBB) has been developed to solve to optimality the HUC with cascading plants. This algorithm is used to solve an MINLP, where the power is a bilinear function of both water head and water flow.

However, in various papers of the literature [16,34,66,70,77,79], the shape of the power function does not correspond to (2.2.1). Indeed, the power function is described as non-convex and non-concave mainly due to turbines efficiency. Turbines efficiency is non-linear with respect to the water flow d_t even with fixed-head $h_t = H$ and this is not captured by equation (2.2.1). Instead of (2.2.1), we consider the following power function, with $\boldsymbol{g}$ being the turbines efficiency.

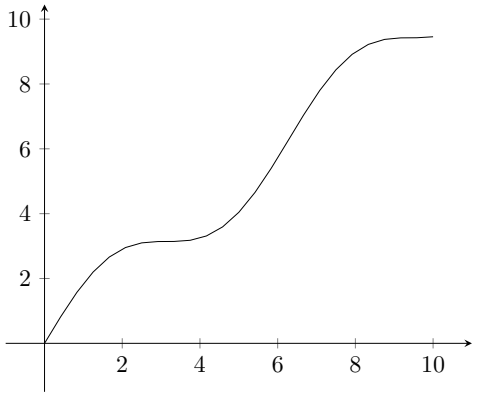
$$\boldsymbol{F}(d_t, h_t) = \rho \cdot G \cdot h_t \cdot \boldsymbol{g}(d_t, h_t) \tag{2.2.2}$$

In the following, we review multiple ways to take into account the turbines efficiency.

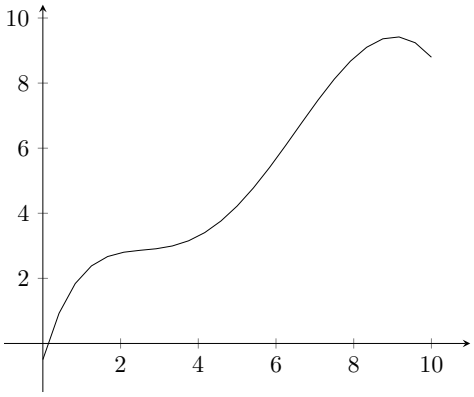
One approach is to rely on polynomial functions, which are well known in the literature of MINLP. In [77] a more sophisticated function than (2.2.1) is provided, where the power is represented by:

$$\boldsymbol{F}(d_t, h_t) = C_1 \cdot (v_t^1)^2 + C_2 \cdot (d_t)^2 + C_3 \cdot v_t^1 \cdot d_t + C_4 \cdot v_t^1 + C_5 \cdot d_t + C_6$$

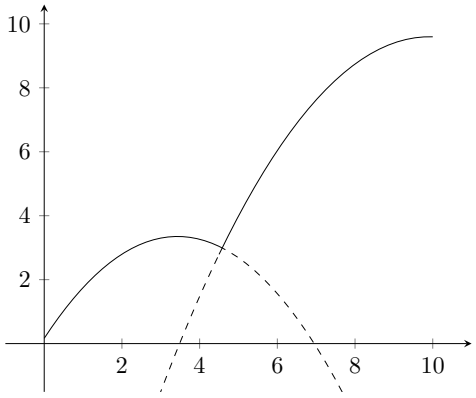
with C_1 to C_6 being constants. This function is a quadratic function composed of a bilinear term and two monomials, depending on the volume and the water flow respectively. The formulation is further



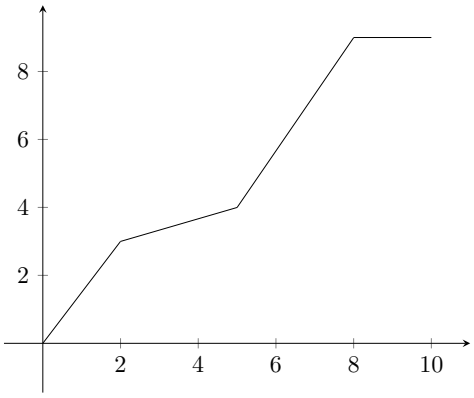
(a) Real-world non-linear function



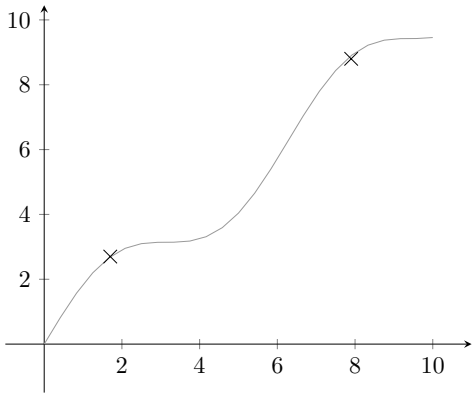
(b) Approximation with a simpler non-linear function



(c) Approximation with a family of elementary functions



(d) Approximation with a PWL function



(e) Approximation with a discrete set of decisions

Figure 2.1: Four different approximations of a real-world non-linear function

enhanced in [96], [40], [42] and [41] where the power function $\boldsymbol{F}(d_t, h_t)$ from [77] depicts the turbine efficiency, rather than the whole power function. In these papers, the whole power function is more complex, and also takes into account other non-linear terms, which we describe later in this review. In particular, the power function in these formulations is represented by polynomials with degrees higher

than 2. Using similar ideas, in [79] an NLP is presented, where the power function is approximated by a family of quadratic functions. More recently, a data-driven study of the non-linearities of the HUC has been conducted [18]. The efficiency function considered is a piecewise polynomial function, where polynomial functions are not necessarily of degree two. In the same category of function, in [4] the power function of a unit features a fraction between a quadratic and a linear function:

$$\boldsymbol{F}(d_t, h_t) = \frac{d_t + C_1 \cdot (v_t^1)^2 + C_2 \cdot v_t^1 + C_3}{C_4 \cdot v_t^1 + C_5}$$

with C_1 to C_5 being constants.

Another approach uses a grid to have a reference for the hydroelectric power function obtained by discretizing the water flow and the head [34]. An algorithm is described in [93] to obtain a set of such grids, each of them representing the power function for a given number of active units. For each point on one of these grids, the value of the hydroelectric power function is computed with a dynamic programming algorithm based on a bilinear function similar to (2.2.1). The resulting grids are overestimation of the power function, meaning that they do not necessarily reflect its actual shape.

Another common modeling alternative from the literature is to approximate the power function with a piecewise linear function (PWL). In [25] and [81], authors introduce a family of univariate PWL functions to model the power depending on the water flow. Each PWL function of this family represents the power with respect to the water flow for a specific volume. The model proposed in [16] for the 1-HUC also expresses the power with a family of univariate PWL functions of the water flow, for specific volumes. Besides, it takes into account the maximum variation of the water flow between two consecutive time periods. An improvement of this model features the rectangle method [32]. The aim is to compute a better approximation when v_t^1 is between two of the specific volumes selected to compute the PWL functions. To do so, the method computes a projection of the power between the two surrounding specific volumes in order to rectify the approximation. There are also iterative methods using PWL functions. In [43] the HUC with cascading plants is considered. It is pointed out that in the fixed-head case, the power depends only on the water flow. Using a PWL function with two pieces, the procedure is to solve a HUC problem with fixed-head iteratively, while updating the head between each iteration until convergence. A variant of the standard PWL models is to consider a PWL function approximating the convex hull of the power function [94]. In such a case, there is a loss of precision in concave parts, but the benefit is that there is no need for any binary variables. Indeed, for any water flow, the corresponding piece of the PWL function is always the one leading to the best value for the objective function. Besides PWL formulations, hyperplanes formulations have also been developed [89]. The aim is to create a set of hyperplanes for each number of active units, in order to linearize the non-linear power function. More precisely, the hyperplanes are deduced from the most efficient point, and each set forms a concave over-estimator of the power function for a given number of active units. As a maximization problem is considered, these hyperplanes yield a convex optimization region. Defining multiple sets aims to produce a more precise approximation, based on the aforementioned grid approximation of the power function for each number of active units [93]. For a given number of active units, the linearization does not require any additional

binary variable. However, binary variables are required to indicate the number of active units at each time period, thus resulting in a MILP. Using the hyperplanes is in practice quite similar to using a PWL function approximating the convex hull of the non-linear function, presented previously for [94].

Comparing modeling alternatives raised interest in recent literature. In [34] three models for the hydro power maintenance scheduling are compared. The three models involve respectively a formulation with hyperplanes [89], a PWL formulation from [32], and a five degree polynomial function. The grid approach [93] is considered as a baseline to compare the economic value of the solutions. The result of this comparison is that the hyperplanes formulation is overall the best compromise between the size and complexity of the optimization problem and the deviation from the reference data. In [96] another comparison of models is provided. The three models considered are a model with high degree polynomials, a standard piecewise linear model, and a piecewise linear model of the convex hull of the power function. The results show that considering the non-linear model yields the best trade-off, as it requires a lower computational time than the standard PWL model, but provides a higher objective value than the convex hull PWL model. Another comparison has been carried out in [40], but instead of comparing modeling alternatives, the following three solution approaches are compared: solving a non-linear HUC with a Lagrangian relaxation, solving a non-linear HUC with the AIMMS outer approximation algorithm, and solving a linearized HUC with a MILP solver. The results show that the first two options yield the best results in terms of objective functions, while solving the MILP model introduces deteriorations of the objective function that vary between 1% and 2 %. However, the computational times of solving the MINLP model are much higher than the other options.

As the focus is to represent the power function, the 1-HUC considered in this chapter is simplified with respect to other components. Thus, many constraints from the literature, listed hereafter, will be ignored. The penstock loss [18] is the power loss due to the friction between the water and the penstock. The spillage [19,70,81] is the process of discharging water from the upstream reservoir to the downstream reservoir without going through the plant. The spilled water does not play a major role in the economic value of the valley. Indeed, it goes directly to the downstream reservoir, without activating any turbine, hence no energy is produced by the spilled water. The main purpose of spilling water is to avoid overfloods. Ramping constraints [32] limit the variation of the water flow between two consecutive time periods. These constraints are used in practice in order to take into account several other uses of water in the valley. In some models, there are start-up and shut-down costs [43,67], making the start-up and shut down of a unit impact the profit. Finally, we consider a predefined unit start-up sequence [34]. This makes it possible to compare models aggregating all units, but also models where each unit is represented individually. Moreover, without such a fixed sequence comparing models would become more difficult, because the efficiency of a turbine can be impacted by their start-up sequence.

When it comes to the head effect, few alternatives are considered in the models of the literature, involving either linear or polynomial functions as described for instance in [4,96,40,42,66,70]. No other modeling alternative is proposed for the head effect.

In this chapter, we compare modeling alternatives, such as in [96] and [34], but not solution techniques as it is done in [40]. To do so, we consider generic models corresponding to the three common

alternatives to represent the power function in the literature, namely a PWL function, a bilinear function or polynomial functions. We also consider in this chapter generic models with non-linear functions that have not yet been studied in the literature of the HUC. As such, we push further the comparison done in [96] and [34], as a larger variety of models is considered. Indeed, LP, MILP, NLP and MINLP models are considered, where some represent the power function of the whole plant, while others represent the power function of each unit explicitly. Besides, all models considered will also be compared for the fixed-head case, a standard 1-HUC simplification.

2.3 Modeling alternatives for the simplified non-linear 1-Hydro Unit Commitment problem

In this section, we first propose a non-linear model capturing all the non-linear characteristics of the 1-HUC^{NL} problem arising from the power generation function. As this first model cannot be solved efficiently, we then present seven modeling alternatives.

2.3.1 The non-linear reference model for the simplified non-linear 1-Hydro Unit Commitment problem

We describe a reference model M_{ref} for the 1-HUC^{NL} problem, specifically designed to encompass all the non-linear characteristics identified in **Table 2.1**. To define M_{ref}, we consider the generic model M_{gen} defined at page 28, from which we specify functions $\boldsymbol{f}$ and $\boldsymbol{F}$.

First, let us focus on function $\boldsymbol{f}$ to define the head. **Figure 2.2a** shows the evolution of the head depending on the volume for a realistic instance named B-T-1, described in **Appendix A.3**. More generally, function $\boldsymbol{f}$ has the following form:

$$h_t = \boldsymbol{f}(v_t^1) = \gamma_1 + \gamma_2 \cdot v_t^1 + \gamma_3 \cdot (v_t^1)^{\gamma_4} \qquad \forall t \leq T \tag{2.3.1}$$

where γ_i are parameters depending on the instance, and $\gamma_4 \in [0.5, 1]$, which means that this function is necessarily concave. Depending on the shape of the reservoir, the function can be quasi linear or have a very noticeable non-linearity, but always stays concave.

Second, let us focus on the power function $\boldsymbol{F}$, featuring the turbine efficiency $\boldsymbol{g}$ as in equation (2.2.2). The only two non-linear terms in the power function are the head h_t and the turbine efficiency $\boldsymbol{g}$. Recall that all non-linear characteristics in **Table 2.1** hold for the fixed-head case. Therefore, these characteristics come from the turbine efficiencies. Hence, we define function $\boldsymbol{g}$ in order for it to correspond to these characteristics. In order to be as close to the physics as possible, we define $\boldsymbol{g}$ as a piecewise non-linear function with K different *five parameter logistic functions* (**5PL**) [48]. **5PL** functions are described in **Appendix A.2**. With K the number of units, the $5PL$ $i \leq K$ represents the efficiency of the first i units combined with respect to the flow d_t. In the general case, function $\boldsymbol{g}$ also depends on the head. In particular, function $\boldsymbol{g}$ for the maximum head is not a linear transformation of function $\boldsymbol{g}$ for the minimum head. First, the turbine efficiency increases non-linearly with respect to the head. Second, with a

larger head, the apex of the efficiency curve is obtained with a higher water flow. Consequently, with a larger head one can use a turbine on a wider range of water flows, meaning that even the water flow d_t to start up the units is larger. To take these effects into account, the parameters of the $\mathbf{5PL}$ functions are dependent on the head h_t.

We define $\boldsymbol{g}$ as a piecewise non-linear function. To do so, we define the following notations:

- a_{it} : the binary variables such that $a_{it} = 1$ if we use the $5PL$ function associated with the first i units at time period t;
- x_{jit} : the continuous variables being the j^{th} parameter of the $5PL$ function for the first i units at time period t;
- α_{ji} and β_{ji} : the constants such that x_{jit} linearly depends on h_t with these parameters.

With these notations, we introduce the following constraints:

$$x_{jit} = \alpha_{ji} + \beta_{ji} \cdot h_t \qquad \forall j \leq 5, \forall i \leq K, \forall t \leq T \tag{2.3.2}$$

$$\sum_{i=1}^{K} a_{it} \leq 1 \qquad \forall t \leq T \tag{2.3.3}$$

$$a_{it} \in \{0,1\} \qquad \forall i \leq K, \forall t \leq T \tag{2.3.4}$$

$$p_t = \boldsymbol{F}(d_t, h_t) = \rho \cdot G \cdot h_t \cdot \sum_{i=1}^{K} a_{it} \cdot \mathbf{5PL}(d_t, x_{1it}, \ldots, x_{5it}) \qquad \forall t \leq T \tag{2.3.5}$$

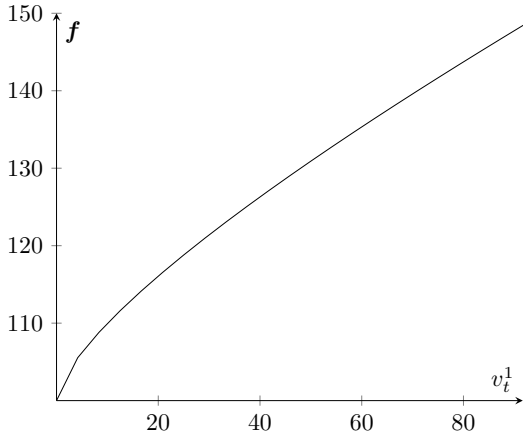
Equalities (2.3.2) define parameters x_{jit} to be linearly dependent on h_t. Constraints (2.3.3) and (2.3.4) ensure that only one $5PL$ function is considered at each time period. Equalities (2.3.5) is the power with $\boldsymbol{g}$ as a piecewise non-linear function.

The complete MINLP model M_{ref} features objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.1)-(2.3.5). The $\mathbf{5PL}$ functions are parameterized such that only the concave part of the $\mathbf{5PL}$ is considered when i units are active, as well as locally linear when a unit starts. Besides, there is a non-differentiable point when starting up a new unit. As the power is clearly non-convex and non-concave, the non-linear power function in M_{ref} features the four main characteristics described in **Table 2.1**.

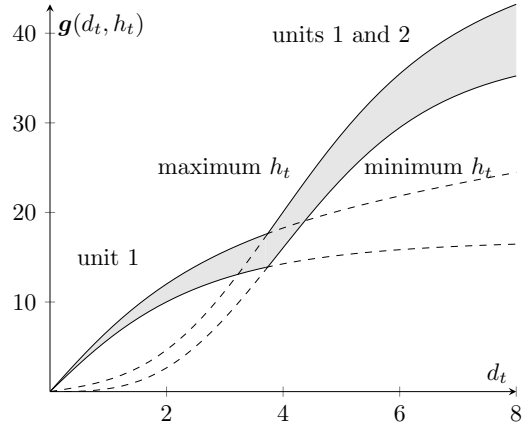
When considering a piecewise (non-)linear model, one usually requires additional constraints to match the value of the binary variables corresponding to the pieces with the value of the decision variables. For model M_{ref}, this means adding constraints to indicate, for a given time period t, which binary variable a_{it} is equal to 1, depending on d_t. However, in some cases, there is no need to describe which binary variable must be equal to 1, and sometimes binary variables are not required such as for convex PWL functions [94]. For model M_{ref}, the $\mathbf{5PL}$ functions are such that, for a given water flow d_t the $\mathbf{5PL}$ function with the highest value represents the power of the units. Because we maximize the profit it turns out that, for a given water flow and if energy prices Λ_t are positive, the $\mathbf{5PL}$ with the highest value

given d_t, will be the one considered for the optimal solution. In this case, the model does not require additional constraints to indicate, for a given t, which one of the variables a_{it} is equal to 1. All of the instances considered in this study satisfy $\Lambda_t \geq 0$, thus there is no need to add constraints specifying which variable a_{it} is equal to 1 for a time period. Conversely, the additional constraints are required for the PWL model shown in **Section 2.3.4**. Note that equation (2.3.3) is still required, otherwise the model could consider multiple active **5*PL*** functions at a given time period, which would induce an incorrect value of the power p_t.

Figure 2.2b shows the evolution of function g with respect to d_t for instance B-T-1 (described in **Table 9** in **Appendix A.3**). The functions in black are for the minimum and the maximum head, and the grey region represents the power function for the possible values of h_t.



(a) Example of function f (constraint (2.3.1))



(b) Example of function g for two units for the minimum and maximum head (constraint (2.3.5))

Figure 2.2: Examples of function f and g

Preliminary computations show that M_{ref} involves higher computational times than any other model presented, and is not practical for most of the instances considered, even for the smallest ones. This is often the case in real world applications where the functions modeling a physical system are either too complex to be implemented or not supported by any solver. Instead of using this model in the comparison, it will be considered as the reference model to benchmark the quality of the modeling alternatives in the experimental results in **Section 2.4**.

Our goal is to derive more tractable models than M_{ref} to capture the non-linearity in the power function. Hence, in the following, we propose modeling alternatives, which do not feature all the characteristics from **Table 2.1**, but can be supported by solvers. The models are described for the 1-HUC^{NL} problem and the fixed-head case.

2.3.2 (Mixed Integer)Non-Linear Program modeling elementary non-linear functions

The models in this section represent the power function of each unit explicitly, in order to have a representation close to the physics. The downside is that for each unit, auxiliary variables are required. Three models are presented. The first one features a family of polynomial functions, the second one a family of **5*PL*** functions with the **max** function, and the last one a family of **5*PL*** functions without the **max** function.

MINLP with a family of polynomial functions: model $M_{2D\text{-}poly}$. The power function of one unit for a given head has a parabolic shape. A parabolic shape can be represented with a quadratic function. Each polynomial function represents the power generated by a unit, plus the contribution of the previous ones, following their startup order. **Figure 2.3** shows an example with two units. We define the following notations:

- b_{it} : the binary variables such that $b_{it} = 1$ if we use the polynomial function of unit i at the time period t;
- y_{qit} : the continuous variables that are the coefficients of monomial d_t^q in the polynomial of unit i at time period t;
- γ_{qi} and δ_{qi} : the constants such that y_{qit} linearly depends on h_t with these parameters.

We introduce the following constraints:

$$y_{qit} = \gamma_{qi} + \delta_{qi} \cdot h_t \qquad \forall q \leq 2, \forall i \leq K, \forall t \leq T \qquad (2.3.6)$$

$$\sum_{i=1}^{K} b_{it} = 1 \qquad \forall t \leq T \qquad (2.3.7)$$

$$b_{it} \in \{0,1\} \qquad \forall i \leq K, \forall t \leq T \qquad (2.3.8)$$

$$p_t = \rho \cdot G \cdot h_t \cdot \sum_{i=1}^{K} b_{it} \cdot \sum_{q=0}^{2} y_{qit} \cdot (d_t)^q \qquad \forall t \leq T \qquad (2.3.9)$$

Equalities (2.3.6) define parameters y_{qit} as linearly dependent on h_t. Constraints (2.3.7) and (2.3.8) ensure that only one of the polynomials is active for each time period. Equalities (2.3.9) express the power with function $\boldsymbol{g}$ represented by a family of polynomial functions.

The complete model, called $M_{2D\text{-}poly}$ is defined by objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.1), (2.3.6)-(2.3.9). It appears that $M_{2D\text{-}poly}$ is a non-convex MINLP as (2.3.1) and (2.3.9) are non-linear. Indeed, function $\boldsymbol{f}$ computing h_t in (2.3.1) is concave, and the polynomial functions in (2.3.9) are concave. But y_{qit} is linear with respect to h_t (2.3.6), and in (2.3.9) variable y_{qit} is multiplied by h_t. So the power function is convex with respect to h_t, as it is an increasing quadratic function. This model represents well the power function, as it takes into account characteristics C1, C3 and C4. However, this model still has downsides, mainly the addition of auxiliary binary variables.

In a similar fashion as for model M_{ref}, model $M_{2D\text{-}poly}$ does not require additional constraints to match the values of b_{it} with the values of d_t. Indeed, for any given water flow d_t, the polynomial with the highest value represents the power of the units. In opposition, one can identify these additional constraints for the PWL model in **Section 2.3.4**. Note that equation (2.3.7) is still required.

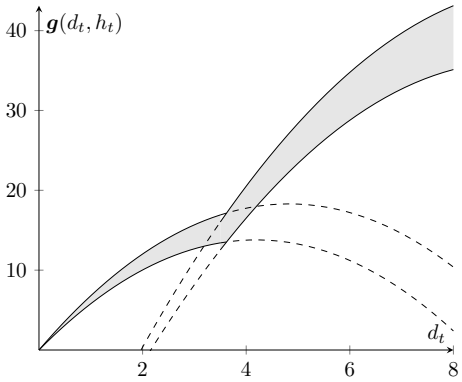


Figure 2.3: Function $\boldsymbol{g}$ with polynomial functions

Model $M_{2D\text{-}poly}$ for the fixed-head 1-HUC^{NL} problem. To adapt the model $M_{2D\text{-}poly}$ to the fixed-head 1-HUC^{NL} problem, with head H, we introduce the following notations:

- Y_{qi} : the constant coefficient of $(d_t)^q$ for unit i.

The power function becomes (2.3.10)

$$p_t = \rho \cdot G \cdot H \cdot \Big(\sum_{i=1}^{K} b_{it} \cdot (\sum_{q=0}^{2} Y_{qi} \cdot (d_t)^q) \Big) \qquad \forall t \leq T \qquad (2.3.10)$$

Model $M_{2D\text{-}poly}$ for the fixed-head 1-HUC^{NL} problem contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.7)-(2.3.8), (2.3.10) and is an MINLP. Note that this model becomes a convex MINLP for the fixed-head 1-HUC^{NL} problem, as all constraints feature concave functions.

NLP with 5*PL* functions using function max: model $M_{5PL\text{-}max}$. Function $\boldsymbol{g}$ can be represented as a sum of **5*PL*** functions [48], where each **5*PL*** represents the power of one unit. By summing properly parameterized **5*PL*** functions, the sum can be a precise approximation of the physical data. **Figure 2.4** shows an example of the sum of **5*PL*** functions as a solid line, and the two separated **5*PL*** as dashed lines.

To represent $\boldsymbol{g}$ with such a sum of **5*PL*** functions, the **5*PL*** functions need to depend on the water flow and the head. As such, these parameters are variables and depend on the unit i and time step t. To use **5*PL*** functions, we introduce the following notations:

- z_{jit} : the continuous variables that are the j^{th} parameter of the **5*PL*** function for the unit i at the time period t;

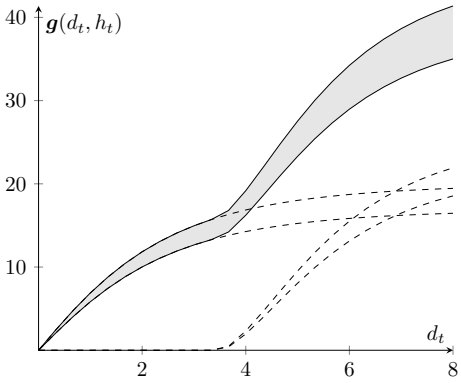


Figure 2.4: Function $\boldsymbol{g}$ with a sum of **5*PL*** functions

- η_{ji} and θ_{ji} : the constants such that z_{jit} linearly depends on h_t with these parameters.

For this model we need the following equalities:

$$z_{jit} = \eta_{ji} + \theta_{ji} \cdot h_t \qquad \forall j \leq 5, \forall i \leq K, \forall t \leq T \tag{2.3.11}$$

$$p_t = \rho \cdot G \cdot h_t \cdot \left(\sum_{i=1}^{K} z_{4it} + \frac{-z_{4it}}{\left(1 + \left(\frac{\mathbf{max}(0, d_t - z_{1it})}{z_{3it}}\right)^{z_{2it}}\right)^{z_{5it}}} \right) \qquad \forall t \leq T \tag{2.3.12}$$

Equalities (2.3.11) set the parameters of the **5*PL*** functions used in function $\boldsymbol{g}$. Equalities (2.3.12) express the power with $\boldsymbol{g}$ as a sum of non-linear functions. These functions are a slight variant of the **5*PL*** functions detailed in **Appendix A.2**. This is because such **5*PL*** functions are not defined when $d_t < z_{1it}$. Consequently, it is necessary to introduce a **max** function in equalities (2.3.12) in order for the **5*PL*** functions to always be defined. Also, we consider parameters η_{3i} and θ_{3i} to be such that $z_{3it} > 0$.

The model $M_{5PL\text{-}max}$ includes objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.1), (2.3.11)-(2.3.12). It is a non-convex non-concave NLP, and the non-linearity takes into account characteristics C1 and C2. As this model contains a **max** function, it is not supported by some global optimization solvers.

Model $M_{5PL\text{-}max}$ for the fixed-head 1-HUC^{NL} problem. To adapt the model $M_{5PL\text{-}max}$ to the fixed-head 1-HUC^{NL} problem, with head H, we introduce the following notations:

- Z_{ji} : the j^{th} parameters for the **5*PL*** function of the unit i.

The power is defined by equalities (2.3.13) with $Z_{3i} > 0$.

$$p_t = \rho \cdot G \cdot H \cdot \left(\sum_{i=1}^{K} Z_{4i} + \frac{-Z_{4i}}{\left(1 + \left(\frac{\mathbf{max}(0, d_t - Z_{1i})}{Z_{3i}}\right)^{Z_{2i}}\right)^{Z_{5i}}} \right) \qquad \forall t \leq T \tag{2.3.13}$$

Model $M_{5PL\text{-}max}$ for the fixed-head 1-HUC^{NL} problem contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.13).

MINLP with 5*PL* functions using auxiliary variables: model $M_{5PL\text{-}bin}$. This model is a variation of $M_{5PL\text{-}max}$, where the **max** is linearized by adding linear constraints and auxiliary variables. We introduce the following notations:

- c_{it} : the binary variables equal to 1 if $d_t \geq z_{1it}$;
- m_{it} : the continuous variables equal to $\max(0, d_t - z_{1it})$.

We introduce the following set of constraints:

$$d_t - z_{1it} \leq m_{it} \leq d_t - z_{1it} + (1 - c_{it}) \cdot (\overline{D} - \underline{Z}_{1it}) \qquad \forall i \leq K, \forall t \leq T \qquad (2.3.14)$$

$$0 \leq m_{it} \leq c_{it} \cdot (\overline{D} - \underline{Z}_{1it}) \qquad \forall i \leq K, \forall t \leq T \qquad (2.3.15)$$

$$c_{it} \in \{0, 1\} \qquad \forall i \leq K, \forall t \leq T \qquad (2.3.16)$$

$$p_t = \rho \cdot G \cdot h_t \cdot \left(\sum_{i=1}^{K} z_{4it} + \frac{-z_{4it}}{\left(1 + \left(\frac{m_{it}}{z_{3it}}\right)^{z_{2it}}\right)^{z_{5it}}} \right) \qquad \forall t \leq T \qquad (2.3.17)$$

Set of constraints (2.3.14)-(2.3.16) ensures $m_{it} = \mathbf{max}(0, d_t - z_{1it})$. Equalities (2.3.17) express the power in the same manner as equalities (2.3.12), but using m_{it}. In a similar manner as for model $M_{5PL\text{-}max}$, we consider parameters η_{3i} and θ_{3i} to be such that $z_{3it} > 0$.

The model $M_{5PL\text{-}bin}$ contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.1), (2.3.11), (2.3.14)-(2.3.17). Unlike the NLP model $M_{5PL\text{-}max}$, model $M_{5PL\text{-}bin}$ is an MINLP, as it requires auxiliary binary variable c_{it}. Model $M_{5PL\text{-}bin}$ can be solved by many more MINLP solvers than model $M_{5PL\text{-}max}$, as function **max** has been linearized. The representation of the power function is the same for $M_{5PL\text{-}max}$ and $M_{5PL\text{-}bin}$, and both models take into account characteristics C1 and C2. Note that the model M_{gen}, with the piecewise non-linear function with **5*PL***, is also an MINLP. The difference is that the binary variables are not the same as the ones in $M_{5PL\text{-}bin}$. Indeed, the binary variables of $M_{5PL\text{-}bin}$ only acts in order to linearize the function **max**, while in M_{gen} they are decision variables.

Model $M_{5PL\text{-}bin}$ for the fixed-head 1-HUC^{NL} problem. To adapt the model $M_{5PL\text{-}bin}$ to the fixed-head 1-HUC^{NL} problem, with head H, we introduce the following notations:

- Z_{ji} : the constants for the parameter j for the **5*PL*** function of the unit i;
- m_{it} : the variables such that $m_{it} = \mathbf{max}(0, d_t - Z_{1i})$.

To ensure the behaviour of variable m_{it}, we add the set of constraints (2.3.18)-(2.3.21), defined as (2.3.14)-(2.3.17), where variables z_{jit} are replaced by constants Z_{ji} for all $t \leq T$, with $Z_{3i} > 0$. Model $M_{5PL\text{-}bin}$ for the fixed-head 1-HUC^{NL} problem contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.28)-(2.31).

2.3.3 (Mixed Integer)Non-Linear Program modeling an aggregated non-linear function

The models introduced in this section represent all units as an aggregated function. The principle is to consider a single function to represent the whole power function, instead of a family or a sum of elementary functions. A single function being less precise, the expected benefit is a quick solution by MINLP tools, as few additional variables and constraints are required. The functions we propose are the following: a polynomial function, a bilinear function, and a finite set of operating flows.

NLP with a high degree polynomial function: model $M_{HD\text{-}poly}$. A model using an aggregated function that represents well the physics is obtained by using a single polynomial function as function $\boldsymbol{g}$. **Figure 2.5** shows an example of an 8$^{\text{th}}$ degree polynomial function for an instance with two units. As $\boldsymbol{g}$ depends on the head h_t, the coefficients of the polynomial are linearly dependent on h_t. We introduce the following notation

- Q : the degree of the polynomial, with $Q = 4 \cdot K$, where K denotes the number of units;
- u_{qt} : the continuous variable that are the coefficient of monomial d_t^q in the polynomial function at time period t;
- μ_q and ν_q : the constants such that u_{qt} linearly depends on h_t with these parameters.

For this model, we need the following equalities:

$$u_{qt} = \mu_q + \nu_q \cdot h_t \qquad \forall q \leq Q, \forall t \leq T \tag{2.3.22}$$

$$p_t = \rho \cdot G \cdot h_t \cdot \Big(\sum_{q=0}^{Q} u_{qt} \cdot (d_t)^q \Big) \qquad \forall t \leq T \tag{2.3.23}$$

Equalities (2.3.22) set the parameters of the polynomial function. Equalities (2.3.23) define the power with $\boldsymbol{g}$ as a polynomial function.

The model, called $M_{HD\text{-}poly}$ includes objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.1), (2.3.22)-(2.3.23). It is an NLP featuring characteristic C1 as it is non-convex and non-concave. The benefits compared to $M_{2D\text{-}poly}$ is that $M_{HD\text{-}poly}$ considers only a single polynomial function. This means that no auxiliary binary variables are required. The downside of $M_{HD\text{-}poly}$ is that high degree polynomials (8 for two units, 20 for five units) can induce large approximation errors. If the water flow can fluctuate between 0 and 100, then it means that the solver might need to compute numbers such as 0.1^8 or 100^8, which are either too small or too large numbers for solvers' precision. Moreover, computational errors can have a dramatic impact for the HUC, as an error for a time period will cumulate and carry over all future time periods [91].

Model $M_{HD\text{-}poly}$ for the fixed-head 1-HUC^{NL} problem. To adapt the model $M_{HD\text{-}poly}$ to the fixed-head 1-HUC^{NL} problem, with head H, we introduce the following notations:

- U_q : the coefficients for the degree q of the polynomial function.

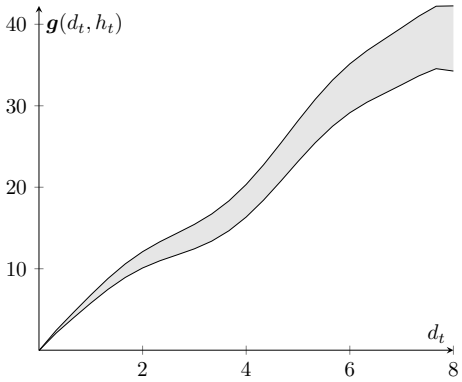


Figure 2.5: Function $\boldsymbol{g}$ with a single polynomial function

The power function becomes:

$$p_t = \rho \cdot G \cdot H \cdot \left(\sum_{q=0}^{Q} U_q \cdot (d_t)^q \right) \qquad \forall t \leq T \qquad (2.3.24)$$

The model $M_{HD\text{-}poly}$ for these special instances contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.24).

NLP with a bilinear function: model M_{bilin}. A type of model often described in the literature to solve the HUC as an MINLP is a bilinear model [70], [19]. **Figure 2.6** shows an example of a two-dimensional projection of a bilinear function. The power is linear with respect to the water flow, and to the head. In the model M_{gen}, the power is already linear with respect to the head. We need to make it also linear with respect to the water flow. To do so, we introduce the following notations:

- ϕ and ψ: the constants such that the power is linearly dependent on the water flow.

We adapt the power function as follows:

$$p_t = \rho \cdot G \cdot h_t \cdot (\phi + \psi \cdot d_t) \qquad \forall t \leq T \qquad (2.3.25)$$

Equalities (2.3.25) express the power as a bilinear function of the head h_t and the water flow d_t.

The model M_{bilin} contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.1), (2.3.25). Equalities (2.3.25) feature a bilinear function of h_t and d_t. A bilinear function is non-convex non-concave even if both variables are positive [84]. However, this function remains simpler than the ones featured in previously described models, such as polynomial or **5*PL*** functions. Besides, unlike $M_{2D\text{-}poly}$, $M_{HD\text{-}poly}$, $M_{5PL\text{-}max}$ or $M_{5PL\text{-}bin}$, model M_{bilin} does not require any additional binary variables. This makes this model a potential candidate to solve quickly the problem. The downside is that this model has the roughest approximation of all models. Indeed, the bilinear function features none of the non-linear characteristics C1, C2, C3 or C4.

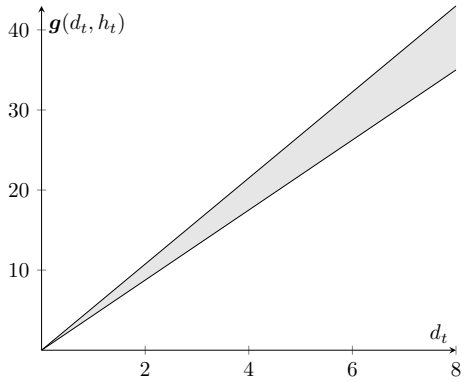


Figure 2.6: Function g with a bilinear function

Model M_{bilin} for the fixed-head 1-HUC^{NL} problem. When we adapt the model M_{bilin} to the fixed-head 1-HUC^{NL} problem, with head H, the model becomes a linear model, where the power is a linear function of the water flow. To do so, we simply adapt the power function as follows:

$$p_t = \rho \cdot G \cdot H \cdot (\phi + \psi \cdot d_t) \qquad \forall t \leq T \tag{2.3.26}$$

The model M_{bilin} for the fixed-head 1-HUC^{NL} problem contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.26), which yields an LP.

MINLP with a discrete set of decisions: model M_{op}. Having a discrete set of decisions for the 1-HUC^{NL} problem means that only a given number of operating flows, say N, are authorized. **Figure 2.7** shows an example of a discrete set of decisions, with $N = 9$. These operating flows are specifically chosen where the power production is the most profitable, and usually are in the concave parts of the original power function in M_{ref}. We introduce the following notations:

- D_i : the constant being the i^{th} operating flow;
- o_{it} : the binary variable such that $o_{it} = 1$ if we use the i^{th} operating flow at time period t.

We consider a model with disjunctive constraints between the operating flows. As such, we need the following constraints:

$$\sum_{i=1}^{N} o_{it} \leq 1 \qquad \forall t \leq T \tag{2.3.27}$$

$$v^1_{t'} = V^1_0 + \sum_{t=1}^{t'} \left(A^1_t - (\sum_{i=1}^{N} o_{it} \cdot D_i) \cdot \Delta \right) \qquad \forall t' \leq T \tag{2.3.28}$$

$$v^2_{t'} = V^2_0 + \sum_{t=1}^{t'} \left(A^2_t + (\sum_{i=1}^{N} o_{it} \cdot D_i) \cdot \Delta \right) \qquad \forall t' \leq T \tag{2.3.29}$$

$$p_t = \rho \cdot G \cdot h_t \cdot \boldsymbol{g}(\sum_{i=1}^{N} o_{it} \cdot D_i, h_t) \qquad \forall t \leq T \tag{2.3.30}$$

Inequalities (2.3.27) ensure that only one operating flow can be active at each time period. The set of equalities (2.3.28)-(2.3.30) corresponds to equalities (2.1.2), (2.1.3) and (2.1.5) from M_{gen}, with operating flows instead of the water flow d_t.

This leads to a new generic model M_{op} containing objective function (2.1.1) and constraints (2.1.6), (2.3.1), (2.3.27)-(2.3.30). Function $\boldsymbol{g}$ in (2.3.30) can be any of the previously described function for models $M_{HD\text{-}poly}$, $M_{2D\text{-}poly}$, $M_{5PL\text{-}max}$, $M_{5PL\text{-}bin}$ or M_{bilin}. Because we have a finite set of operating flows, function $\boldsymbol{g}$ for M_{op} will feature none of the characteristics C1 to C4, regardless of the function considered. Model M_{op} can be beneficial because its solution space is drastically smaller than the others but does not offer as much freedom, in particular when target volumes occur. As the operating flows are amongst the most profitable ones, the solution might still be close to the optimal solution. The downside of this model is that target volumes can be unreachable with the chosen set of operating flows, thus leading to infeasibility.

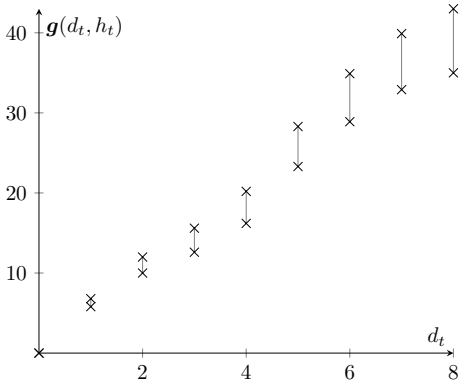


Figure 2.7: Function $\boldsymbol{g}$ with a discrete set of decisions

Model M_{op} for the fixed-head 1-HUC^{NL} problem. When we adapt the model M_{op} for the fixed-head 1-HUC^{NL} problem, with head H, the model becomes a MILP model. In this case the power only depends on the water flow, thus there is a finite set of possible powers. As such, the model becomes a MILP as we have to choose a pair (operating flow, power produced) amongst a list of pairs at each time period. We introduce the following notations:

- P_i : the constant being the power generated for the i^{th} operating flow.

The model is very similar to M_{op} for the general 1-HUC^{NL} problem, but we define the power differently as we use constants P_i:

$$p_t = \sum_{i=1}^{N} o_{it} \cdot P_i \qquad \forall t \leq T \tag{2.3.31}$$

The model M_{op} adapted for the fixed-head 1-HUC^{NL} problem contains objective function (2.1.1) and constraints (2.1.6), (2.3.27)-(2.3.29), (2.3.31). We can notice that this model contains very few variables, as only the decision variables o_{it} are required.

2.3.4 Mixed Integer Linear Program modeling piecewise linear functions

MILP model with a piecewise linear function: model M_{PWL}. The model of this section follows the common practice of using a PWL approximation when modeling a non-linear expression. **Figure 2.8** shows an example of a piecewise linear function. Our comparisons of a PWL model with the previously described non-linear models are mostly focused on the precision, i.e., the quality of the solutions obtained. As such, we will consider a standard PWL formulation [27] [45], more precisely the convex combination formulation. There exist much more efficient formulations, e.g. the logarithmic formulation in [104], but they will not be considered as it will not impact the value of the solution, but only the computational time.

A generic way to obtain a two-dimensional PWL function is to use the one-dimensional method described in [32]. It is a generalization of the convex combination formulation [45]. The method described to approximate a non-linear function $\boldsymbol{f}(x,y)$ is as follows. We fix B_x variables on the x axis, $(\widetilde{x}_1,..,\widetilde{x}_{B_x})$, and B_y variables on the y axis $(\widetilde{y}_1,..,\widetilde{y}_{B_y})$. For each $\widetilde{y}_j$, we approximate $f(x,\widetilde{y}_j)$ with a PWL function $\boldsymbol{l}(x,\widetilde{y}_j)$, where each $\widetilde{x}_i$, $i \leq B_x$ acts like a break point. It means that piece i of $\boldsymbol{l}(x,y_j)$ is a linear function between $\widetilde{x}_i$ and $\widetilde{x}_{i+1}$. We obtain then B_y PWL functions with $B_x - 1$ pieces. The value for $\boldsymbol{l}(x,y)$, $y \in [\widetilde{y}_j,\widetilde{y}_{j+1}]$, is approximated by $\boldsymbol{l}(x,\widetilde{y}_j)$. For the 1-HUC^{NL} problem, we will approximate the power function with respect to the water flow d_t for a set of fixed volumes $\widetilde{y}_j$, $j \in 1,...,B_y$.

In this model, we aggregate both non-linear functions $\boldsymbol{f}$ and $\boldsymbol{g}$ as a unique function to represent the power. To do so, it is possible to replace h_t by $f(v_t^1)$ in equalities (2.1.5) from M_{gen}. Thus, the power is defined as follows, and we only need to approximate one two-dimensional non-linear function for the whole model:

$$p_t = \boldsymbol{F}(d_t, v_t^1) = \rho \cdot G \cdot \boldsymbol{f}(v_t^1) \cdot \boldsymbol{g}(d_t, \boldsymbol{f}(v_t^1))$$

To use the PWL approximation, we introduce the following notations:

- $\boldsymbol{l}(v_t^1, d_t)$: the two dimensional PWL approximation of $\boldsymbol{F}(v_t^1, d_t)$;
- B_y : the number of one dimensional PWL functions allocated on the volume axis;
- B_x : the number of breakpoints on the water flow axis;
- $\tilde{V}_j^1$: the volume corresponding to the j^{th} one dimensional PWL function;
- $\tilde{D}_i$: the water flow at breakpoint i for any one dimensional PWL functions.

And we include the following variables:

- l_{jt} : the value $\boldsymbol{l}(\tilde{V}_j^1, d_t)$ of the one dimensional PWL function j at time period t;
- r_{it} : the binary variables such that $r_{it} = 1$ if d_t is located on the interval $[\tilde{D}_i, \tilde{D}_{i+1}]$;
- w_{it} : the continuous variables such that d_t is the convex combination $w_{it}\tilde{D}_i + w_{i+1t}\tilde{D}_{i+1}$;
- s_{jt} : the binary variables such that $s_{jt} = 1$ if v_t^1 is located in the interval $[\tilde{V}_j^1, \tilde{V}_{j+1}^1]$.

The PWL formulation requires the following constraints:

$$\sum_{i=1}^{B_x} r_{it} = 1 \qquad \forall t \leq T \qquad (2.3.32)$$

$$w_{it} \leq r_{i-1t} + r_{it} \qquad \forall i \leq B_x, \forall t \leq T \qquad (2.3.33)$$

$$\sum_{i=1}^{B_x} w_{it} = 1 \qquad \forall t \leq T \qquad (2.3.34)$$

$$d_t = \sum_{i=1}^{B_x} w_{it} \cdot \tilde{D}_i \qquad \forall t \leq T \qquad (2.3.35)$$

$$l_{jt} = \sum_{i=1}^{B_x} w_{it} \cdot \boldsymbol{F}(\tilde{V}_j^1, \tilde{D}_i) \qquad \forall j \leq B_y, \forall t \leq T \qquad (2.3.36)$$

$$r_{it} \in \{0, 1\} \qquad \forall i \leq B_x, \forall t \leq T \qquad (2.3.37)$$

$$0 \leq w_{it} \leq 1 \qquad \forall i \leq B_x, \forall t \leq T \qquad (2.3.38)$$

$$\sum_{j=1}^{B_y} s_{jt} \cdot \tilde{V}_j^1 \leq v_t^1 \leq \sum_{j=1}^{B_y} s_{jt} \cdot \tilde{V}_{j+1}^1 \qquad \forall t \leq T \qquad (2.3.39)$$

$$\sum_{j=1}^{B_y} s_{jt} = 1 \qquad \forall t \leq T \qquad (2.3.40)$$

$$l_{jt} - \overline{P}_t \cdot (1 - s_{jt}) \leq p_t \leq l_{jt} + \overline{P}_t \cdot (1 - s_{jt}) \qquad \forall j \leq B_y, \forall t \leq T \qquad (2.3.41)$$

$$s_{jt} \in \{0, 1\} \qquad \forall j \leq B_y, \forall t \leq T \qquad (2.3.42)$$

Constraints (2.3.32)-(2.3.38) are the standard convex combination formulation for a one-dimensional PWL function, applied to approximate function $\boldsymbol{F}$ for each given volume. These constraints ensure that l_{jt} is the value, at time period t of the PWL function approximating $\boldsymbol{F}$ for volume $\tilde{V}_j^1$. Equalities (2.3.32) express that exactly one variable r_{it} is equal to one at time period t, meaning that we consider one piece of the PWL function at time period t. Inequalities (2.3.33) allow the weight r_{it} of a breakpoint to be greater than zero only if one for the two surrounding pieces is considered at time period t. Equalities (2.3.34)-(2.3.35) ensure that d_t is the convex combination of the $\tilde{D}_i$ with the weights r_{it} at time period t. Equalities (2.3.36) define the PWL approximation of the power function. Constraints (2.3.37)-(2.3.38) give the domain of variables r_{it} and w_{it}.

We have described the constraints to obtain $l_{t,j}$ the approximated value of $\boldsymbol{F}$ at time period t using univariate PWL functions. Now we need constraints (2.3.39)-(2.3.42) in order to obtain the power from the value $l_{t,j}$ for the fixed volume $\tilde{V}_j^1$, with $\tilde{V}_j^1 \leq v_t^1 \leq \tilde{V}_{j+1}^1$. Constraints (2.3.39)-(2.3.40) ensure $s_{jt} = 1$ if $v_t^1 \in [\tilde{V}_j^1; \tilde{V}_{j+1}^1]$, and exactly one variable s_{jt} is equal to 1. Variable s_{jt} then indicates which PWL function should be considered depending on the volume. Inequalities (2.3.41) ensure $p_t = l_{jt}$ if $s_{jt} = 1$, or give trivial bounds if $s_{jt} = 0$. Constraints (2.3.42) provide the domain of s_{jt}.

The MILP model M_{PWL} contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.32)-(2.3.42). The consequences of this model being a MILP are twofold. On the one hand, it can be solved with powerful MILP tools. On the other hand it includes a lot of auxiliary variables and constraints, and it does not include any of the non-linear characteristics of the power function.

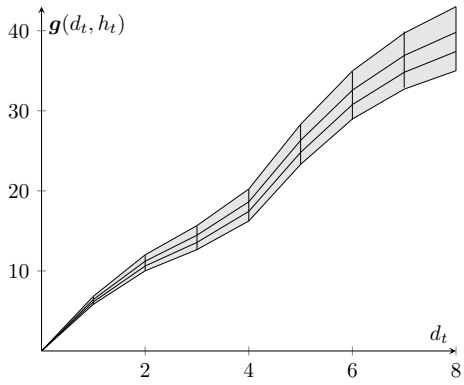


Figure 2.8: Function $\boldsymbol{g}$ with a piecewise-linear function

Model M_{PWL} for the fixed-head 1-HUC^{NL} problem. When we adapt the model M_{PWL} to the fixed-head 1-HUC^{NL} problem, with head H, we approximate a one-dimensional power function. To do so, we use the convex combination formulation [45], which is the formulation generalized for model M_{PWL} in the general case. The convex combination formulation adapted for the fixed-head 1-HUC^{NL} problem requires constraints (2.3.32)-(2.3.35) and (2.3.37)-(2.3.38). We express the power as follows:

$$p_t = \sum_{i=1}^{B_x} w_{it} \cdot \rho \cdot G \cdot H \cdot \boldsymbol{g}(\tilde{D}_i, H) \qquad \forall t \leq T \qquad (2.3.43)$$

Thus the model M_{PWL} for these special instances contains objective function (2.1.1) and constraints (2.1.2)-(2.1.3), (2.1.6)-(2.1.7), (2.3.32)-(2.3.35),(2.3.37)-(2.3.38), (2.3.43).

2.3.5 Summary of models and non-linear functions

We have described a total of fourteen different models. All models have the same objective function (2.1.1). Also, most of the models share the same set of constraints. We define constraint sets S1=(2.1.2)-

(2.1.3),(2.1.6)-(2.1.7),(2.3.1) and S2=(2.1.2)-(2.1.3),(2.1.6)-(2.1.7). **Table 2.2** summarizes all models with their constraints. The difference between these models is the representation of the power function.

Table 2.2: Summary of the proposed models

Model	Section	General cases	Fixed-head cases
$M_{2D\text{-}poly}$	2.3.2	S1,(2.3.6)-(2.3.9)	S2,(2.3.7)-(2.3.8),(2.3.10)
$M_{5PL\text{-}max}$	2.3.2	S1,(2.3.11)-(2.3.12)	S2,(2.3.13)
$M_{5PL\text{-}bin}$	2.3.2	S1,(2.3.11),(2.3.14)-(2.3.17)	S2,2.28-2.31
$M_{HD\text{-}poly}$	2.3.3	S1,(2.3.22)-(2.3.23)	S2,(2.3.24)
M_{bilin}	2.3.3	S1,(2.3.25)	S2,(2.3.26)
M_{op}	2.3.3	(2.1.1),(2.1.6),(2.3.1),(2.3.27)-(2.3.30)	(2.1.1),(2.1.6), (2.3.27)-(2.3.29),(2.3.31)
M_{PWL}	2.3.4	S2,(2.3.32)-(2.3.42)	S2,(2.3.32)-(2.3.35),(2.3.37)-2.3.38,2.3.43

Table 2.3 shows the type of program for each model. Also, in the column approx. char., a check (resp. cross) indicates that the model properly does (resp. not) approximate the characteristic of the original power function. From a theoretical point of view, none of the presented models perfectly fits the power function of M_{ref}. Indeed, none of the models features all four non-linear characteristics of the power functions. However, it will be shown in the numerical experiments that some models allow for very small approximation errors, while other models, with simpler linear or non-linear expressions, lead to shorter computational times. **Table 2.4** shows the size of each model, namely the number of constraints (#cst), the number of binary and continuous variables (respectively #b-var and #c-bar). We recall that K is the number of units, T the number of time periods, Q the degree of the polynomial function (model $M_{HD\text{-}poly}$), B_x the number of breakpoints of the PWL functions and B_y the number of PWL functions (model M_{PWL}).

Table 2.3: Comparison of the models non-linear characteristics

Model	1-HUC^{NL} problem	Fixed-head 1-HUC^{NL} problem	Approx. Char.			
			C1	C2	C3	C4
$M_{2D\text{-}poly}$	non-convex MINLP	convex MINLP	✓	✗	✓	✓
$M_{5PL\text{-}max}$	non-convex NLP	non-convex NLP	✓	✓	✗	✗
$M_{5PL\text{-}bin}$	non-convex MINLP	non-convex MINLP	✓	✓	✗	✗
$M_{HD\text{-}poly}$	non-convex NLP	non-convex NLP	✓	✗	✓	✗
M_{bilin}	non-convex NLP	LP	✗	✗	✗	✗
M_{op}	non-convex MINLP	MILP	✗	✗	✗	✗
M_{PWL}	MILP	MILP	✗	✗	✗	✗

It is also possible to compare the models, on the basis of the difficulty for the solvers to manage their non-linear expression. A way to measure this is to compare the size of the reformulation binary tree for the non-linear expressions as in [95]. The reformulation binary tree is a tree where each node is an operation and each leaf is a variable. As such, a larger reformulation binary tree often means a harder to manage non-linear function by non-linear solvers. Following this metric, ***5PL*** functions are by far

Table 2.4: Number of constraints, binary variables and continuous variables for each model

	1-HUC^{NL} problem			Fixed-head 1-HUC^{NL} problem		
Model	#cst	#b-var	#c-var	#cst	#b-var	#c-var
$M_{2D\text{-}poly}$	$(11+3\cdot K)\cdot T$	$K\cdot T$	$(5+3\cdot K)\cdot T$	$10\cdot T$	$K\cdot T$	$4\cdot T$
$M_{5PL\text{-}max}$	$(10+5\cdot K)\cdot T$	0	$(5+5\cdot K)\cdot T$	$9\cdot T$	0	$4\cdot T$
$M_{5PL\text{-}bin}$	$(10+15\cdot K)\cdot T$	$K\cdot T$	$(5+10\cdot K)\cdot T$	$(9+10\cdot K)\cdot T$	$K\cdot T$	$(4+5\cdot K)\cdot T$
$M_{HD\text{-}poly}$	$(10+Q)\cdot T$	0	$(5+Q)\cdot T$	$9\cdot T$	0	$4\cdot T$
M_{bilin}	$10\cdot T$	0	$5\cdot T$	$9\cdot T$	0	$4\cdot T$
M_{op}	$9\cdot T$	$L\cdot T$	$5\cdot T$	$8\cdot T$	$L\cdot T$	$4\cdot T$
M_{PWL}	$(14+4\cdot B_x+4\cdot B_y)\cdot T$	$(B_x+B_y)\cdot T$	$(4+B_x+B_y)\cdot T$	$(12+4\cdot B_x)\cdot T$	$B_x\cdot T$	$(4+B_x)\cdot T$

the most difficult functions, followed by high degree polynomials, two degree polynomials, bilinear functions and linear functions.

Recall that the power function of model $M_{5PL\text{-}max}$ features a max function. Consequently, the power function for $M_{5PL\text{-}max}$ is nonsmooth and with discontinuous derivatives. This means that this model is not supported by all non-linear solvers.

2.4 Numerical experiments

The computational evaluation is performed via Neos Server [28] on machine prod-exec-7 in October and November 2021 using the following five MINLP solvers: ANTIGONE, BARON, COUENNE, LINDOGlobal, SCIP, along with the MILP solver CPLEX. For MINLP solvers, the GAMS format is used for input files, while for the MILP solver, the LP format is used. All experiments are performed on a single machine, with a 2x Intel Xeon Gold 5218 @ 2.3GHz processor with 384 GB of RAM, using a single thread. The computational time limit is set to 10800 seconds.

2.4.1 Instances, parameters, terminology and metrics

Parameters of the original power function The numerical values for the parameters of the power functions featured in the different models are obtained by fitting their power functions to the one of M_{ref}. The fitting is done via Scipy's curve_fit function[1], using non-linear least squares. This approach does not provide an a priori precision for the resulting function with respect to the data. Recall that the purpose of this work is to study and analyse various approximations of the power function. Thus the parameters of the head functions of all proposed models are the ones of M_{ref}.

Parameters of model M_{PWL} For model M_{PWL} we also want to take into account the impact of the number of linear pieces, therefore we will define three variants of M_{PWL}: $M_{PWL\text{-}1}$, $M_{PWL\text{-}2}$ and $M_{PWL\text{-}3}$, respectively with B_x and B_y equal to 5, 20 and 100. For every variant, the breakpoints are defined as

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html, accessed: 2023-01-09

equidistant instead of being tailored to each instance. Preliminary results show that model M_{PWL} is not significantly penalized in terms of approximation error compared to the other models, with equidistant breakpoints. Hence, we will not investigate the best distribution of the breakpoints, in order to keep a simple model for each modeling alternative.

Parameters of model M_{op} For model M_{op}, a discrete set of decision variables is to be chosen. For the considered instances, we define $\boldsymbol{g}$ as the non-linear function of model $M_{5PL\text{-}bin}$, and we consider 5 operating flows per unit, i.e., $N = 5 \cdot K$. We will not consider models with more operating flows, as the model contains $\mathbf{5}\boldsymbol{PL}$ functions that are already difficult to handle. Additional operating flows would make the model irrelevant as it would become too hard to solve.

Variable bounds All models contain variables that are subject to an equality constraint such as the head h_t or power p_t. These variables have physical bounds, but these are not expressed explicitly in the models, as they are set through the equality constraints. However, when using global solvers, it is a good practice to bound every variable. Hence, for the experiments, every variable has an upper and lower bound, even if these bounds are trivially satisfied through the equality constraints. Note also that these bounds could be improved, but this requires a complete work which is out of the scope of this study. Moreover, solvers usually implement bound tightening techniques. As we also compare the solvers for the proposed model, we do not implement extra features (reformulation, bound tightening etc...). As such the solvers are compared taking into account their complete sets of tools.

Instances All instances considered are derived from parameter sets A and B, detailed in **Appendix A.3**. These parameter sets are inspired from real instances from EDF. Different variants of these sets are created to form a larger set of instances. The varying non-linear features and the corresponding parameters of the 1-HUC^{NL} problem are as follows. The *size of the instance* varies with the number of time periods. *Equality constraints* appear as soon as target volumes are accounted for in the instance. Two features of the non-linear function can be changed: the *number of inflection points* and the *degree of non-linearity*. These features are respectively linked to the number of units, and to when the transition from a unit to another occurs when increasing or decreasing the water flow. The last feature is the *sensitivity of the decision variables*, which measures how much the decision can affect the dynamical behavior of the physical system. For the 1-HUC^{NL} problem, the smaller the water flows are relative to the absolute volume, the less the volumes change over the time periods. In **Appendix A.3**, the features are further described, and an equation for the sensitivity of the decision variables is provided. Also, **Table A.2** summarizes the instances and their features.

Terminology, notations and metrics We introduce additional terminology to compare the different models on the considered instances. For the comparisons, we also define the metrics used and their notation.

A *configuration* is defined as a pair (instance, model). For all models, a configuration is solved to optimality when the optimality gap between the primal and dual values is below 0.1%. Note that the optimality gap is not computed in the exact same manner for every solver, but remains very similar. The *optimal solution of a configuration* is the solution when the configuration is solved to optimality. The value of a configuration is the value of its optimal solution. The *recalculated value of a solution* is the value of the solution, evaluated with the non-linear functions of M_{ref}. Such value is obtained as follows. Consider the water flows of a solution. The recalculated value of this solution is the value of the objective function of M_{ref} with the same water flows. The *recalculated value of a configuration* is the recalculated value of the optimal solution of the configuration. A configuration is *solvable* by a solver if the solver supports the model. A configuration is *solved* if it is solved to optimality with at least one solver within the time limit. A configuration is *feasible* with a solver when it is not solved to optimality, but a solution is found within the time limit. A configuration is *infeasible* with a solver if the solver proves the configuration to have no feasible solution within the time limit.

The metrics used to compare the models and the solvers are as follows. The *computational time* (CT) of a configuration is the time required to return the optimal solution. The *approximation error* (AE) of a configuration is the relative difference between the value of the optimal solution of the configuration, and the recalculated value of the configuration. The *distance to the best recalculated value* (DB) of a configuration is the relative difference between the recalculated value of the configuration, and the highest recalculated value of all configurations with the same instance.

As specified, configurations are solved with several solvers. We define the *virtual best solver* (VBS) [59] of a given configuration as the solver that requires minimal CT to solve the configuration. Results show that the AE (resp. the DB) of a configuration is the same for every solver. Thus, for our results, the VBS is the solver that has the configuration solved to optimality in minimal CT. For our analysis we use the metrics of the configurations with their VBS. All figures and tables for the results are with the VBS, except for **Tables 2.8** and **2.9** that display the results for each solver. Note that some configurations are not solvable with every solver. Indeed, model $M_{5PL\text{-}max}$ is only supported by LINDOGlobal and SCIP.

2.4.2 Model comparison

In this section, we present the results of the model comparison.

Results summary To summarize the results, **Figure 2.9** shows a bargraph which represents two categories of results. First, the height of the bar for a model corresponds to the proportion of configurations solved with their VBS. Second, the lightest color shows for a model the proportion of configurations for which the model has the lowest DB compared to the other models. Similarly, the second and third lightest color for a model correspond to the proportion of configurations for which the model has the second and third lowest DB. The darkest color for a model corresponds to the proportion of configura-

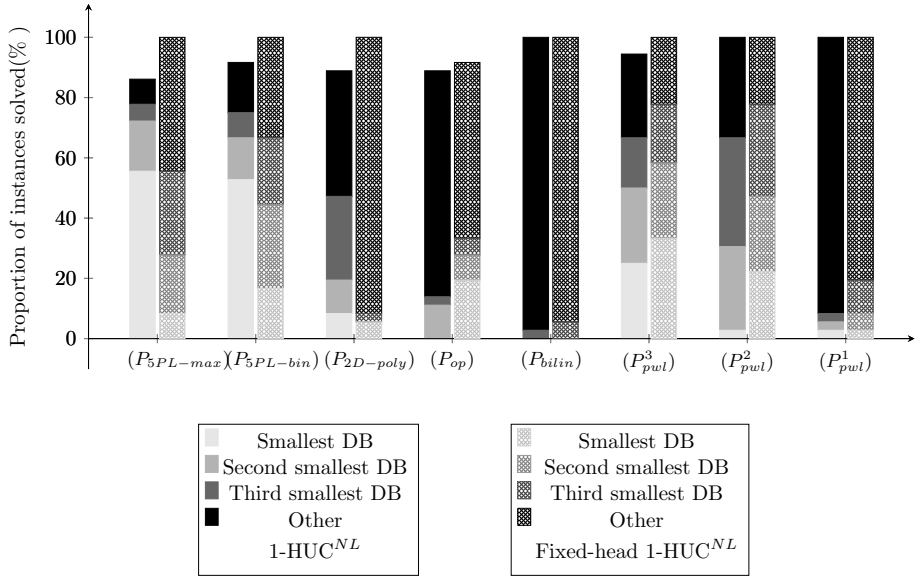


Figure 2.9: Proportion of configurations for each model solved by their VBS

tions for which the model does not yield a DB which is amongst the three lowest DB. These results are distinguished for both the 1-HUC^{NL} problem and the fixed-head 1-HUC^{NL} problem.

In the remainder of this section, we highlight key observations of the model comparison for both 1-HUC^{NL} problems, with and without a fixed-head.

Finding 1: Infeasibility of model $M_{HD\text{-}poly}$. None of the configurations with $M_{HD\text{-}poly}$ returns a feasible solution.

Model $M_{HD\text{-}poly}$ contains high degree polynomials which can yield very large and small numbers. This produces floating point errors for the solvers, which makes this model impractical without a dedicated data pre-processing.

It follows that results related to model $M_{HD\text{-}poly}$ are not enclosed.

Finding 2: Models M_{op}, $M_{2D\text{-}poly}$ can yield infeasibilities. **Table 2.5** shows, for each model, the proportion of solved (%solved), feasible (%feasible), and infeasible (%infeasible) configurations with their VBS. Note that there is no case where the status is undefined: for every configuration, either a feasible solution is found, or the infeasibility is proven within three hours.

In the case of M_{op} infeasibilities can happen because there are instances with target volumes that cannot be reached with the finite set of operating flows. In the case of $M_{2D\text{-}poly}$, infeasibilities occurs for instances of the 1-HUC^{NL} problem with high degree of non-linearity. However, it is not clear why such configurations are deemed infeasible by the solvers.

Finding 3: Considering a fixed-head leads to reduced CT. **Figure 2.10** shows on the y-axis the proportion of configurations solved with their VBS, under a CT threshold given on the x-axis. The configu-

Table 2.5: Proportion of solution status returned for the configurations for each model by their VBS

	1-HUC^{NL}			Fixed-head 1-HUC^{NL}		
Model	%solved	%feasible	%infeasible	%solved	%feasible	%infeasible
$M_{5PL\text{-}max}$	86.1	13.9	0.0	100.0	0.0	0.0
$M_{5PL\text{-}bin}$	91.7	8.3	0.0	100.0	0.0	0.0
$M_{2D\text{-}poly}$	88.9	0.0	11.1	100.0	0.0	0.0
M_{op}	88.9	2.8	8.3	91.7	0.0	8.3
M_{bilin}	100.0	0.0	0.0	100.0	0.0	0.0
$M_{PWL\text{-}3}$	94.4	5.6	0.0	100.0	0.0	0.0
$M_{PWL\text{-}2}$	100.0	0.0	0.0	100.0	0.0	0.0
$M_{PWL\text{-}1}$	100.0	0.0	0.0	100.0	0.0	0.0

rations are color-coded depending on the model of the configuration.

Clearly, for every model, the CT is reduced when solving the fixed-head 1-HUC^{NL} problem.

There are multiple reasons why the CT is lower for the fixed-head case. First, only a one dimensional non-linearity is considered, in opposition to a two-dimensional one and a single-dimensional one in the general case. Also, for all models by M_{bilin}, fewer variables are required when the head is fixed. Then, models M_{op}, M_{bilin} have a linear relaxation in the fixed-head case, whereas these model have a non-linear relaxation in the general case.

Finding 4: Considering a fixed-head leads to increased AE, yielding similar AE for all models. **Figure 2.11** shows on the y-axis the proportion of configurations solved with their VBS, under an AE threshold given on the x-axis. The configurations are color-coded depending on the model of the configuration.

It appears that the AE increases for each model in the fixed-head case. In addition, all models then yield a very similar AE.

We can explain the increased AE for the fixed-head case as follows. In practice, when the volume changes, then the head also changes. However this is not captured when the fixed-head is considered. As such, a fixed-head model will induce higher AE independently of the model selected. This is why all models yield high and similar AE.

Finding 5: Models M_{bilin}, $M_{PWL\text{-}2}$, $M_{2D\text{-}poly}$ and $M_{5PL\text{-}bin}$ are the most efficient ones for the 1-HUC^{NL} problem. **Table 2.6** shows for each model, the average CT and AE for both the 1-HUC^{NL} problem and the fixed-head 1-HUC^{NL} problem. When calculating the average CT, a CT of 10800 seconds (the time limit) is considered for configurations that are not solved to optimality. Infeasible configurations are not taken into account for this average. **Figure 2.12** depicts the average CT and AE for each model, for the 1-HUC^{NL} problem with and without fixed-head.

When considering the CT and the AE metrics as two criterias, models M_{bilin}, $M_{PWL\text{-}2}$, $M_{2D\text{-}poly}$ and $M_{5PL\text{-}bin}$ are the most efficient ones. Indeed, for any other model, one of the four cited models has a lower CT and a lower AE. We cannot deduce the best overall model, as it depends on the user needs.

Finding 6: Models $M_{PWL\text{-}2}$, $M_{2D\text{-}poly}$, M_{op} and $M_{5PL\text{-}max}$ are the most efficient ones for the fixed–head 1-HUC^{NL} problem. In a similar fashion as for the previous finding, we detect models $M_{PWL\text{-}2}$, $M_{2D\text{-}poly}$, M_{op} and $M_{5PL\text{-}max}$ to be the most efficient ones. Note that models $M_{PWL\text{-}2}$, $M_{2D\text{-}poly}$ are amongst the most efficient for the 1-HUC^{NL} problem with and without a fixed-head.

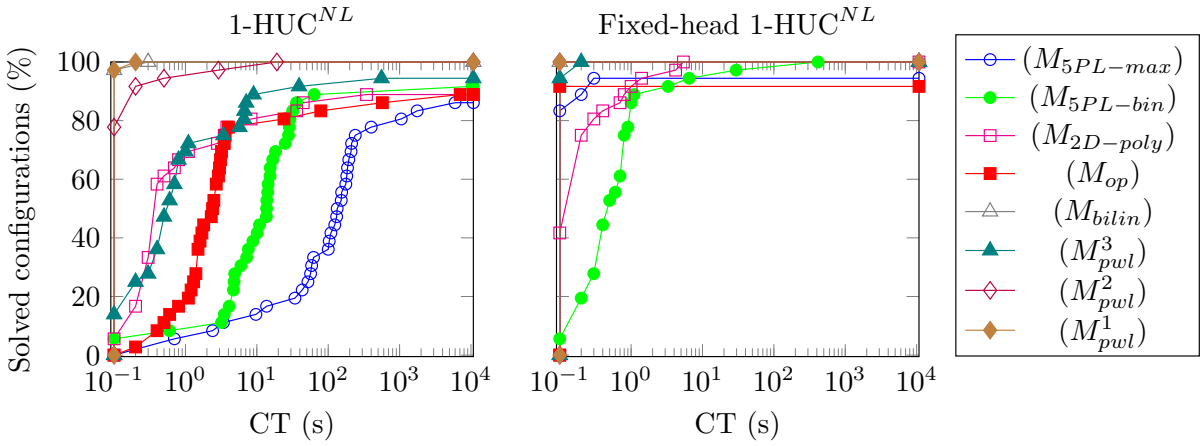


Figure 2.10: Proportion of configurations solved with their VBS where the CT is under a CT threshold

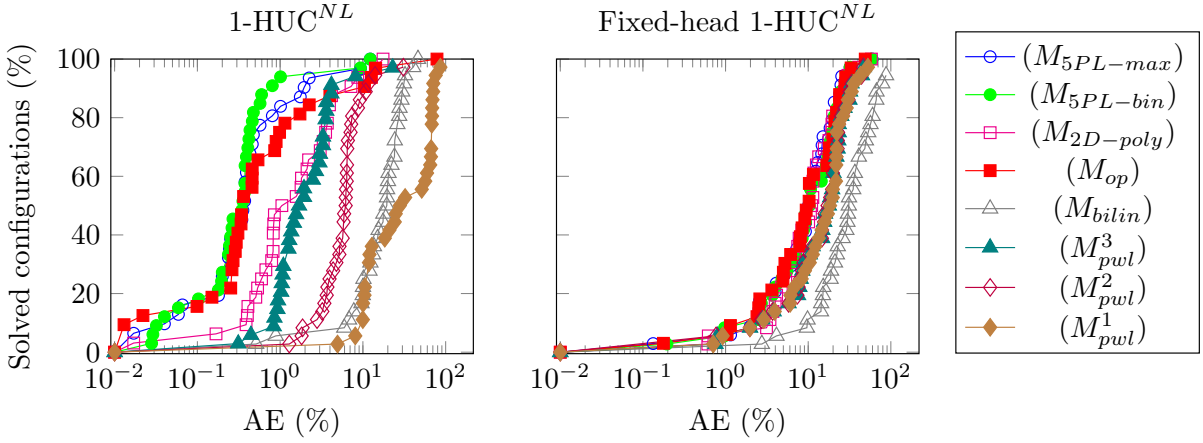


Figure 2.11: Proportion of configurations solved with their VBS where the AE is under an AE threshold

Table 2.6: average CT and AE for each model with its VBS, and CT and AE trade-off for each pair of models for the 1-HUC^{NL} problem

	model	$M_{5PL\text{-}max}$	$M_{5PL\text{-}bin}$	$M_{2D\text{-}poly}$	M_{op}	M_{bilin}	$M_{PWL\text{-}3}$	$M_{PWL\text{-}2}$	$M_{PWL\text{-}1}$
1-HUC^{NL}	average CT	1832.1	1200.6	12.6	520.1	0.1	618.4	0.7	0.1
	average AE	1.16	0.96	2.88	4.16	19.14	3.43	7.33	40.26
fixed-head 1-HUC^{NL}	average CT	600.6	13.1	0.5	3.3	0.1	0.1	0.1	0.1
	average AE	12.90	13.85	14.11	13.43	37.92	18.34	18.14	18.93

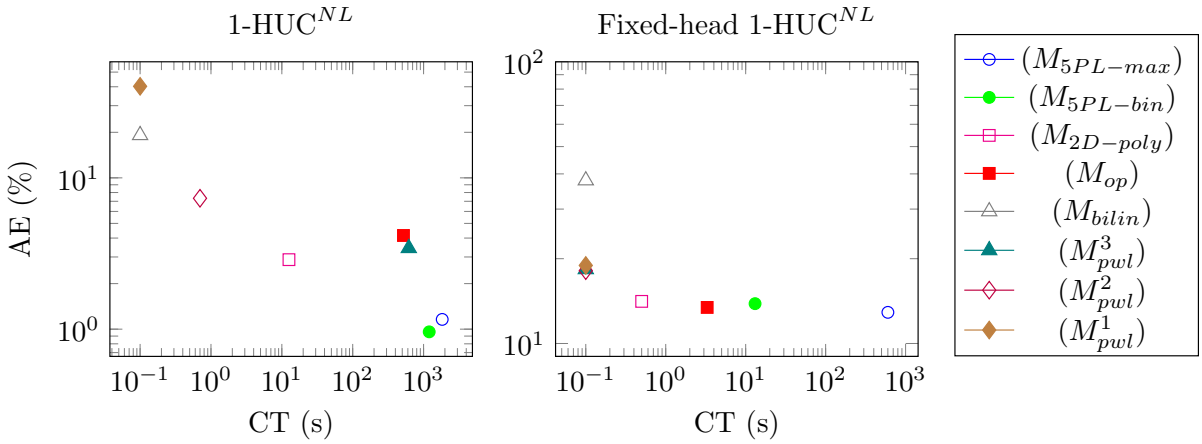


Figure 2.12: Average CT and AE trade-off for each model

Finding 7: Metrics DB and AE are correlated. **Figure 2.13** shows on the y-axis the proportion of configurations solved with their VBS, under a DB threshold given on the x-axis.

Results show that $M_{5PL\text{-}bin}$ and $M_{5PL\text{-}max}$ solve more than 60% of the 1-HUC^{NL} problem instances with a DB below 0.01%. In opposition, models $M_{PWL\text{-}1}$ and M_{bilin} yield a DB above 1% for nearly all instances. Besides, models with the smallest DB tend to be the ones with smaller AE. The only exception is model M_{op} with high DB despite a low AE.

The reason why the DB is correlated to the AE is because models with lower AE correspond more precisely to the physics. Hence, the solutions obtained with these models are closer to the real optimal solution of M_{ref} than the solutions obtained with models with a higher AE. Model M_{op} is an exception. Indeed, model M_{op} has a similar AE as $M_{5PL\text{-}max}$ and $M_{5PL\text{-}bin}$ as shown in **Figure 2.11**. However the DB for M_{op} is much higher than for $M_{5PL\text{-}max}$ and $M_{5PL\text{-}bin}$ (**Figure 2.13**), which means that its solutions are of lesser economic quality. This is because for M_{op}, there is a finite set of operating flows, and every solution must have water flows within this set. It is possible that the optimal solutions obtained with M_{op} are far from the optimal solutions of M_{ref} for which the water flows are not restricted to a finite set.

Impact of each characteristics We now summarize the impact of modifying the characteristics of the 1-HUC^{NL} problem. In **Table 2.7**, we present the general impact for the 1-HUC^{NL} problem when modifying one characteristic. One arrow up (resp. down) means a moderate increase (resp. decrease), and two arrows up (resp. down) means a large increase (resp. decrease). We also added remarks for some models. The results presented here are further described in **Appendix A.4**, and the corresponding tables are shown in **Appendix A.5**.

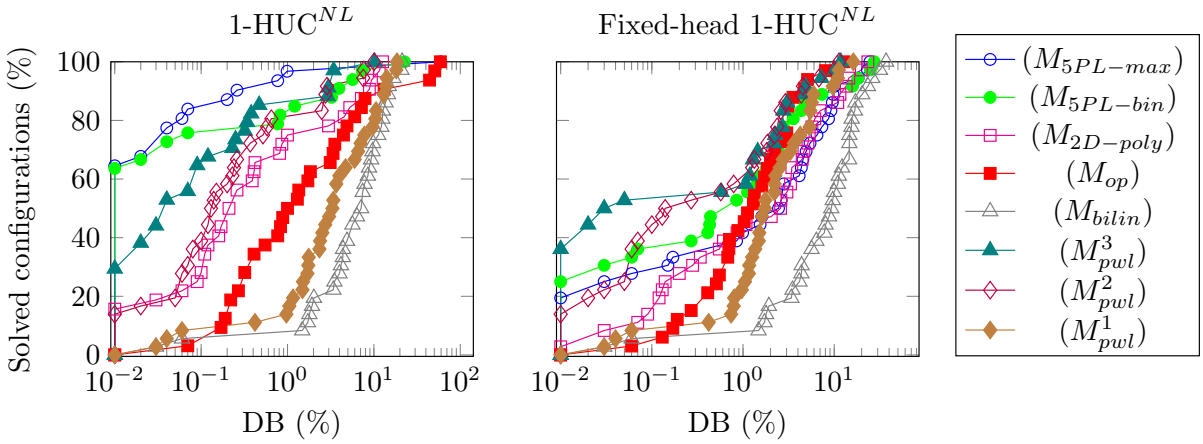


Figure 2.13: Proportion of configurations solved with their VBS where the DB is under a DB threshold

Table 2.7: Summary of the impact of each characteristic

modified characteristics	1-HUC^{NL}	fixed-head 1-HUC^{NL}	remarks
increased size	CT↑↑	AE↑↑	CT increases much more for slower models
added equality constraints	CT↓, AE↓↓	AE↓↓	M_{op} can yield infeasibilities
increased degree of non-linearity	AE↑↑	AE↑↑	$M_{2D\text{-}poly}$ can yield infeasibilities
increased number of inflection points	CT↑	CT↑	CT increases only for $M_{2D\text{-}poly}$, $M_{5PL\text{-}bin}$ and $M_{5PL\text{-}max}$
decreased sensitivity to decision variables	CT↓	AE↓↓	all variants of M_{PWL} can yield high AE

2.4.3 Solver comparison

Previous results are presented with respect to the *virtual best solver* (VBS). However, in a practical case it may not be convenient to use the VBS, as it could be difficult to have access to as many solvers. In this section we will analyze the behaviour of each solver independently. As aforementioned, a solved configuration has, for every solver, the same AE. Only the proportion of configurations solved and the CT can change from a solver to another. Hence, we do not consider metric AE when comparing the solvers. In the tables presented in this section, the following notations are used:

- %S: proportion of configurations solved;
- avg-CT: average CT in seconds, for solved configurations;
- NS: model not supported by the solver;
- NR: model supported by the solver, but experiments are not reported.

For a model, a solver dominates another solver if it has a higher %S, and a smaller avg-CT.

Finding 1: The performance of a solver highly depends on the model. **Table 2.8** and **Table 2.9** show for each model the proportion of configurations solved with each solver, and the average CT, respectively

for the 1-HUC^{NL} problem and for the fixed-head 1-HUC^{NL} problem. For each model, the smallest avg-CT and the highest %S are emphasized in bold. If for a model, the two metrics are in bold for a solver, then it dominates every other solver for the model.

The results confirm that the performance of a solver highly depends on the model. For instance, when the 1-HUC^{NL} problem is considered (**Table 2.8**), solver SCIP dominates solver ANTIGONE for model $M_{5PL\text{-}bin}$, but ANTIGONE dominates SCIP for model $M_{2D\text{-}poly}$.

Table 2.8: Proportion of configurations for each model solved by each solver and related average CT for the 1-HUC^{NL} problem

	ANTIGONE		BARON		COUENNE		LINDOGlobal		SCIP		CPLEX	
Model	%S	avg-CT	%S	avg-CT	%S	avg-CT	%S	avg-CT	%S	avg-CT	%S	avg-CT
$M_{5PL\text{-}max}$	NS		NS		NS		38.9	616.36	86.1	606.87	NS	
$M_{5PL\text{-}bin}$	19.4	824.84	88.9	25.63	69.4	163.23	86.1	446.92	88.9	218.00	NS	
$M_{2D\text{-}poly}$	44.4	2.51	88.9	31.32	63.9	91.27	80.6	30.36	19.4	828.72	NS	
M_{op}	50.0	235.24	83.3	258.94	52.8	439.83	75.0	375.10	69.4	44.23	NS	
M_{bilin}	75.0	0.10	100.0	0.08	97.2	0.25	75.0	4.72	100.0	0.17	NS	
$M_{PWL\text{-}3}$	NR		NR		NR		NR		NR		94.4	19.07
$M_{PWL\text{-}2}$	NR		NR		NR		NR		NR		100.0	0.71
$M_{PWL\text{-}1}$	NR		NR		NR		NR		NR		100.0	0.02

Table 2.9: Proportion of configurations for each model solved by each solver and related average CT for the fixed-head 1-HUC^{NL} problem

	ANTIGONE		BARON		COUENNE		LINDOGlobal		SCIP		CPLEX	
Model	%S	avg-CT	%S	avg-CT	%S	avg-CT	%S	avg-CT	%S	avg-CT	%S	avg-CT
$M_{5PL\text{-}max}$	NS		NS		NS		97.2	159.96	94.4	0.13	NS	
$M_{5PL\text{-}bin}$	30.6	856.07	100.0	13.13	100.0	109.84	61.1	58.80	100.0	2.95	NS	
$M_{2D\text{-}poly}$	72.2	0.15	100.0	0.51	100.0	0.87	100.0	1.36	100.0	100.31	NS	
M_{op}	NR		NR		NR		NR		NR		91.7	0.01
M_{bilin}	NR		NR		NR		NR		NR		100.0	0.00
$M_{PWL\text{-}3}$	NR		NR		NR		NR		NR		100.0	0.05
$M_{PWL\text{-}2}$	NR		NR		NR		NR		NR		100.0	0.02
$M_{PWL\text{-}1}$	NR		NR		NR		NR		NR		100.0	0.01

Finding 2: Solvers SCIP and BARON are the most efficient ones for (MI)NLP models. **Tables 2.10** and **2.11** show the proportion of instances where a solver is the VBS, for each model.

Solver SCIP is the most efficient for model $M_{5PL\text{-}max}$. Solver BARON is the most efficient for any other non-linear model, namely models $M_{5PL\text{-}bin}$, $M_{2D\text{-}poly}$, M_{bilin} and M_{op} for the 1-HUC^{NL} problem and models $M_{5PL\text{-}bin}$ and $M_{2D\text{-}poly}$ for the fixed-head 1-HUC^{NL} problem.

We distinguish two exceptions where solver ANTIGONE is the most efficient for model M_{bilin} for the 1-HUC^{NL} problem and model $M_{2D\text{-}poly}$ for the fixed-head 1-HUC^{NL} problem. However, **Tables 2.8** and

2.9 show that in these cases, the problem is solved very quickly. Thus it is possible that ANTIGONE is quicker than BARON only on easy configurations, and may only be quicker to startup.

Finding 3: (MI)LP models are efficiently solved by CPLEX. All linear models, namely models $M_{PWL\text{-}1}$, $M_{PWL\text{-}2}$ and $M_{PWL\text{-}3}$ for the 1-HUC^{NL} problem and models M_{bilin}, M_{op} for the fixed-head 1-HUC^{NL} problem are efficiently solved by CPLEX.

Tables 2.8 and **2.9** show that most linear models are solved on average in less than 1 second. Besides, when M_{op} is infeasible, it is also proven infeasible by CPLEX within one second.

Table 2.10: Proportion of configurations for each model where a solver is the VBS for the 1-HUC^{NL} problem

Model	ANTIGONE	BARON	COUENNE	LINDOGlobal	SCIP	CPLEX
$M_{5PL\text{-}max}$	NS	NS	NS	16.1	83.9	NS
$M_{5PL\text{-}bin}$	0.0	54.6	30.3	15.2	0.0	NS
$M_{2D\text{-}poly}$	12.5	84.4	3.1	0.0	0.0	NS
M_{op}	0.0	90.6	0.0	6.3	3.1	NS
M_{bilin}	63.9	33.3	0.0	0.0	2.8	NS
$M_{PWL\text{-}3}$	NR	NR	NR	NR	NR	100.0
$M_{PWL\text{-}2}$	NR	NR	NR	NR	NR	100.0
$M_{PWL\text{-}1}$	NR	NR	NR	NR	NR	100.0

Table 2.11: Proportion of configurations for each model where a solver is the VBS for the fixed-head 1-HUC^{NL} problem

Model	ANTIGONE	BARON	COUENNE	LINDOGlobal	SCIP	CPLEX
$M_{5PL\text{-}max}$	NS	NS	NS	5.6	94.4	NS
$M_{5PL\text{-}bin}$	2.8	72.2	0.0	13.9	11.1	NS
$M_{2D\text{-}poly}$	52.8	33.3	0.0	13.9	0.0	NS
M_{op}	NR	NR	NR	NR	NR	100.0
M_{bilin}	NR	NR	NR	NR	NR	100.0
$M_{PWL\text{-}3}$	NR	NR	NR	NR	NR	100.0
$M_{PWL\text{-}2}$	NR	NR	NR	NR	NR	100.0
$M_{PWL\text{-}1}$	NR	NR	NR	NR	NR	100.0

In **Appendix A.4** an analysis of the impact of each feature of a 1-HUC^{NL} problem instance is described.

2.4.4 General modeling recommendations derived from the numerical experiments

The results show that the choice of the model highly depends on the needs of a user. Indeed, if a low AE is required, models $M_{5PL\text{-}bin}$, and $M_{5PL\text{-}max}$ are the most efficient. However, not all solvers support $M_{5PL\text{-}max}$, and larger CT can be induced, meaning that $M_{5PL\text{-}bin}$ is overall a better alternative than

$M_{5PL\text{-}max}$. For the fixed-head 1-HUC^{NL} problem, models M_{op} and $M_{2D\text{-}poly}$ can also be considered, as they yield nearly the same AE as $M_{5PL\text{-}bin}$, and $M_{5PL\text{-}max}$. In opposition, if one requires low computational times, models M_{bilin}, and $M_{PWL\text{-}1}$ are the most suitable. However, the AE can be very high with such models. For a more balanced option, three types of models stand out: M_{PWL} and $M_{2D\text{-}poly}$ for the 1-HUC^{NL} problem and M_{PWL}, M_{op} and $M_{2D\text{-}poly}$ for the fixed-head 1-HUC^{NL} problem. We list them hereafter giving for each of them their main strengths and weaknesses. Firstly, M_{PWL} usually provides a good trade-off between CT and AE. However the proper number of pieces cannot always be deduced in advance. Consequently, a trial and error procedure may be necessary to determine a piecewise linear function with a good trade-off. Secondly, model M_{op} can lead to the smallest AE, and it can be solved faster than the sophisticated models $M_{5PL\text{-}bin}$ and $M_{5PL\text{-}max}$. The drawback is that in the case of an instance with equality constraints on the water resource, there may not be a feasible solution for model M_{op}. Thirdly, model $M_{2D\text{-}poly}$ also yields a very good trade-off between AE and CT. However, for some configurations featuring model $M_{2D\text{-}poly}$, all solvers fail to find a solution, even if there is a feasible solution with the model. This illustrates the intrinsic difficulties of the current solvers for some non-linear models (see also the case of $M_{HD\text{-}poly}$ described in **Section 2.4.2**).

The choice of the solver impacts the CT and the proportion of instances solved. The results indicate that BARON is the most efficient non-linear solver when the model is supported, otherwise SCIP is the most efficient one. For the three balanced models highlighted, solver BARON is the most efficient for the non-linear ones (model M_{op} for the 1-HUC^{NL} problem and $M_{2D\text{-}poly}$ in any case), and a specialized MILP solver should be considered for the linear ones (model M_{op} for the fixed-head 1-HUC^{NL} problem and M_{PWL} in any case).

2.5 Conclusion

In this chapter, we compared modeling alternatives to solve the non-linear 1-HUC problem. The focus of this study is the power function, that is a two-dimensional non-convex and non-concave function of the water flow and the reservoir volume. Seven alternatives are considered, covering a wide variety of family of models, including models from the literature and also new models. The comparison of these alternatives is conducted on several sets of instances, each of them focusing on a particular characteristic of the 1-HUC^{NL} problem. Non-linear modeling alternatives are solved using five different non-linear solvers, in order to identify pairs of model/solver yielding the best results.

From this study, four out of the seven models stand-out: M_{bilin}, M_{PWL}, $M_{2D\text{-}poly}$ and $M_{5PL\text{-}max}$. Similarly, three models stand out in the fixed-head case: M_{PWL}, M_{op} and $M_{2D\text{-}poly}$. Indeed, these models give the best trade-offs between metrics such as the computational time, the precision and the feasibility. For each of these models, we also recommend one solver which yields the smallest computational times.

In the next chapter, we will select one of the highlighted models and extend it to integrate hydraulic constraints. The selected model will be enhanced, through a reformulation and a bound-tightening

procedure. Finally, we will define the resulting variants of the 1-Hydro Unit Commitment problem that will be considered for the remainder of the thesis, as well as their mathematical models.

3 Selected model for the remainder of this thesis: a discretized fixed head model

Table of contents

In the previous chapter, we compared seven modeling alternatives to represent a simplified non-linear 1-Hydro Unit Commitment problem. From this comparison, we highlighted the most efficient alternatives in the general and the fixed-head case. For the remainder of the thesis, we plan to develop solution approaches for the 1-Hydro Unit Commitment problem, focusing on hydraulic constraints. For this purpose, we start from one of the previously highlighted models, and define generalized versions of it that includes these hydraulic constraints.

This chapter comprises three sections. In **Section 3.1** we present the modeling alternative selected and the integration of hydraulic constraints. In **Section 3.2** we propose reformulations and bound-tightening techniques to improve the model. In **Section 3.3** we specify the resulting mathematical models that will be studied in the subsequent chapters of this thesis, before the conclusion.

3.1 Selected power function modeling and integration of hydraulic constraints

The remainder of the thesis will focus on hydraulic constraints. These constraints include the minimum and maximum bounds on the volumes, featured in some instances of the previous chapter, but

also ramping and min-up/down constraints which where not included previously. As the head effect is not related to these constraints, we will consider the fixed-head 1-HUC problem. The previous study has shown that M_{PWL}, $M_{2D\text{-}poly}$ and M_{op} are the best models for the fixed-head 1-HUC with regards to the non-linear power function. Among these models, the one yielding the smallest approximation errors is M_{op}. Therefore our goal for the remainder of thesis will be to focus on models resulting from the integration of hydraulic constraints into model M_{op}, with the ambition of obtaining high-performance algorithms. In addition, for the previous study we limited the number of operating flows to 5 for model M_{op}. This is because model M_{op} is non-linear for the general case, and more operating flows lead to very high computational times. However, for the fixed-head case, model M_{op} becomes linear, and one can consider more than 5 operating points without inducing excessive computational times. Defining a larger set of operating flows also increases the quality of the solutions obtained, and reduces the chances of M_{op} to yield infeasibilities. Finally, the current approach at EDF is to solve a model similar to M_{op} with a fixed-head, which is another reason to study further this modeling alternative in the thesis.

The straightforward integration of pumps as well as ramping and min-up/down constraints into M_{op} leads to model (3.1.1)-(3.1.15). To that end we define an operating point as a pair (D_i, P_i), with D_i a water flow, and P_i the associated power considering a fixed head. The plant operates on N turbining points (indexed $\{1,\dots,N\}$), M pumping points (indexed $\{-1,\dots,-M\}$) and an idle operating point (indexed 0). With $\mathcal{N} = \{-M,\dots,0,\dots,N\}$, we introduce $x_{t,i}$ the binary variable indicating whether the plant is at least at operating point $i \in \mathcal{N}$ at time period $t \leq T$ and binary variable $v_{t,i}$ indicating the start-up of an operating point i at time period t. We also recall that R_u (resp. R_d) is the maximum increase (resp. decrease) of the ramping constraints, and L is the duration of the min-up/down constraints. The resulting model is as follows:

$$\max \sum_{t=1}^{T}\sum_{i\in\mathcal{N}} \Lambda_t P_i x_{t,i} + \Phi^1\Big(\sum_{t=1}^{T}(A_t^1 - \sum_{i\in\mathcal{N}} D_i x_{t,i})\Big) + \Phi^2\Big(\sum_{t=1}^{T}(A_t^2 + \sum_{i\in\mathcal{N}} D_i x_{t,i})\Big) \tag{3.1.1}$$

$$\text{s.c.} \quad V_0^1 + \sum_{t=1}^{t'}(A_t^1 - \sum_{i\in\mathcal{N}} D_i x_{t,i}) \leq \overline{V}_{t'}^1, \qquad \forall t' \leq T \tag{3.1.2}$$

$$V_0^1 + \sum_{t=1}^{t'}(A_t^1 - \sum_{i\in\mathcal{N}} D_i x_{t,i}) \geq \underline{V}_{t'}^1, \qquad \forall t' \leq T \tag{3.1.3}$$

$$V_0^2 + \sum_{t=1}^{t'}(A_t^2 + \sum_{i\in\mathcal{N}} D_i x_{t,i}) \leq \overline{V}_{t'}^2, \qquad \forall t' \leq T \tag{3.1.4}$$

$$V_0^2 + \sum_{t=1}^{t'}(A_t^2 + \sum_{i\in\mathcal{N}} D_i x_{t,i}) \geq \underline{V}_{t'}^2, \qquad \forall t' \leq T \tag{3.1.5}$$

$$x_{t,i} \geq x_{t,i+1}, \qquad \forall t \leq T, \forall i \in \{0,\dots,N-1\} \tag{3.1.6}$$

$$x_{t,i} \geq x_{t,i-1}, \qquad \forall t \leq T, \forall i \in \{-M+1,\dots,0\} \tag{3.1.7}$$

$$x_{t,1} + x_{t,-1} \leq 1, \qquad \forall t \leq T \tag{3.1.8}$$

$$\sum_{i \in \mathcal{N}} D_i x_{t,i} - \sum_{i \in \mathcal{N}} D_i x_{t-1,i} \leq R_u, \qquad \forall t \in \{2, \dots, T\} \tag{3.1.9}$$

$$\sum_{i \in \mathcal{N}} D_i x_{t-1,i} - \sum_{i \in \mathcal{N}} D_i x_{t,i} \leq R_d, \qquad \forall t \in \{2, \dots, T\} \tag{3.1.10}$$

$$\sum_{t'=t-L+1}^{t} v_{t',i} \leq x_{t,i} \qquad \forall t \in \{L, \dots, T\}, \forall i \in \mathcal{N} \tag{3.1.11}$$

$$\sum_{t'=t-L+1}^{t} v_{t',i} \leq 1 - x_{t-L,i} \qquad \forall t \in \{L, \dots, T\}, \forall i \in \mathcal{N} \tag{3.1.12}$$

$$v_{t,i} \geq x_{t,i} - x_{t-1,i} \qquad \forall t \in \{2, \dots, T\}, \forall i \in \mathcal{N} \tag{3.1.13}$$

$$x_{t,i} \in \{0, 1\}, \qquad \forall t \leq T, \forall i \in \mathcal{N} \tag{3.1.14}$$

$$v_{t,i} \in \{0, 1\}, \qquad \forall 2 \leq t \leq T, \forall i \in \mathcal{N} \tag{3.1.15}$$

The objective function (3.1.1) maximizes the value of the valley, taking into account the value of the remaining water at time period T, and the value of the power produced. Inequalities (3.1.2) to (3.1.5) are the upper and lower bounds of the upstream and downstream reservoirs. Inequalities (3.1.6) and (3.1.7) express the cumulative nature of the operating points. Inequalities (3.1.8) forbid the units to pump and turbine simultaneously. Inequalities (3.1.9) and (3.1.10) are the ramping constraints. Inequalities (3.1.11) to (3.1.13) are the min-up/down constraints as formulated in [85] with additional binary variables $v_{t,i}$. Inequalities (3.1.14) and (3.1.15) express the binary nature of variables $x_{t,i}$ and variables $v_{t,i}$.

Operating points are modeled in a cumulative fashion by inequalities (3.1.6) and (3.1.7). It is also possible to model operating points with disjunctive constraints. Even if these two alternatives do not yield the same relaxation [101] in general, for linear models such as the one presented in this section, they are equivalent [27]. Hence, we consider in the cumulative operating points, in order to remain closer to the current model at EDF.

3.2 Reformulation and improvements

In this section, we reformulate the previously defined models in order for them to be easier to exploit. Then, we present an improvement of the model based on bound tightenings.

3.2.1 Shifting all operating points

It is convenient to only have operating points with non-negative power and flow. In [3] a modification on the flows and the volume bounds is defined in order to only have such operating points. First, each turbining operating point i, for $i \in \{1, .., N\}$ is renumbered $i + M$, yielding operating point (D'_{i+M}, P'_{i+M}) with $D'_{i+M} = D_i$ and $P'_{i+M} = P_i$. For each pumping operating point $i \in \{-M, \dots, -1\}$, a turbining operating

point numbered $i+M+1$ is created, yielding operating point (D'_{i+M+1}, P'_{i+M+1}) with $D'_{i+M+1} = |D_i|$ and $P'_{i+M+1} = |P_i|$. Operating point 0 remains unchanged. As such, all the operating points have non-negative cumulated flows. For a given $t \leq T$, the bounds on the volume $\overline{V}^1_t$ and $\underline{V}^1_t$ (resp. $\overline{V}^2_t$ and $\underline{V}^2_t$) are shifted in order to keep the same feasible solutions: $\overline{V'}^1_t = \overline{V}^1_t - t\sum_{i=-1}^{-M}|D_i|$ and $\underline{V'}^1_t = \underline{V}^1_t - t\sum_{i=-1}^{-M}|D_i|$ (resp. $\overline{V'}^2_t = \overline{V}^2_t + t\sum_{i=-1}^{-M}|D_i|$ and $\underline{V'}^2_t = \underline{V}^2_t + t\sum_{i=-1}^{-M}|D_i|$).

Example 2

Consider the following instance of the fixed-head 1-HUC problem. The initial volumes are $V^1_0 = 50$, $V^2_0 = 30$. *With* $T = 3$, *upstream reservoir bounds are* $\overline{V}^1 = [100, 100, 20]$ *and* $\underline{V}^1 = [0, 0, 20]$, *and downstream reservoir bounds are* $\overline{V}^2 = [60, 60, 60]$ *and* $\underline{V}^2 = [0, 0, 0]$. *Consider the associated model* (3.1.1)-(3.1.15) *with* $N = 3$ *turbining operating points* $(D_1 = 3, P_1 = 3)$, $(D_2 = 4, P_2 = 3)$, $(D_3 = 2, P_3 = 2)$ *and* $M = 2$ *pumping operating points* $(D_{-1} = -3, P_{-1} = -5)$, $(D_{-2} = -2, P_{-2} = -4)$

The renumbering is such that the 3 turbining operating points, of index 1 *to* 3 *are now of index* $1+M$ *to* $3+M$, *i.e.,* $(D'_3 = 3, P'_3 = 3)$, $(D'_4 = 4, P'_4 = 3)$ *and* $(D'_5 = 2, P'_5 = 2)$ *From pumping operating point* -1 *a turbining operating point* $M - 1 + 1 = 2$ *is created, and similarly from the pumping operating point* -2 *operating point* $M - 2 + 1 = 1$ *is created, i.e.,* $(D'_1 = 2, P'_1 = 4)$ *and* $(D'_2 = 3, P'_2 = 5)$.

The upstream reservoir upper bounds are modified as follows $\overline{V'}^1_1 = \overline{V}^1_1 - 1 \cdot (2+3) = 95$, $\overline{V'}^1_2 = \overline{V}^1_2 - 2 \cdot (2+3) = 90$, $\overline{V'}^1_3 = \overline{V}^1_3 - 3 \cdot (2+3) = 5$. *Similarly, we obtain the following upstream reservoir lower bounds* $\underline{V'}^1 = [-5, -10, 5]$. *The shift is done in the opposite way for the downstream reservoir,* $\overline{V'}^2 = [65, 70, 75]$ *and* $\underline{V'}^2 = [5, 10, 15]$.

As a consequence, in the remainder of the thesis, we consider without loss of generality a plant featuring only turbines, meaning that $\mathcal{N} = \{0, \ldots, N\}$. In this case, constraints (3.1.7) and (3.1.8) are not necessary as there are no pumps, and no renumbering is required.

3.2.2 Rewriting the resource windows and the objective function

Rewriting the resource windows. We focus on the resource windows on the volume, represented with (3.1.2), (3.1.3), (3.1.4) and (3.1.5). These constraints can be rewritten as follows:

$$\sum_{t=1}^{t'}\sum_{i=1}^{N} D_i x_{t,i} \geq V^1_0 + \sum_{t=1}^{t'} A^1_t - \overline{V'}^1_{t'}, \quad \forall t' \leq T \tag{3.2.1}$$

$$\sum_{t=1}^{t'}\sum_{i=1}^{N} D_i x_{t,i} \leq V^1_0 + \sum_{t=1}^{t'} A^1_t - \underline{V'}^1_{t'}, \quad \forall t' \leq T \tag{3.2.2}$$

$$\sum_{t=1}^{t'}\sum_{i=1}^{N} D_i x_{t,i} \leq \overline{V'}^2_{t'} - V^2_0 - \sum_{t=1}^{t'} A^2_t, \quad \forall t' \leq T \tag{3.2.3}$$

$$\sum_{t=1}^{t'}\sum_{i=1}^{N} D_i x_{t,i} \geq \underline{V'}^2_{t'} - V^2_0 - \sum_{t=1}^{t'} A^2_t, \quad \forall t' \leq T \tag{3.2.4}$$

There are redundancies between (3.2.1) and (3.2.4), and between (3.2.2) and (3.2.3). Let us introduce bounds $\beta_{t'}$ and $\alpha_{t'}$ in the following way with $t' \leq T$:

$$\beta_{t'} = \max(V_0^1 + \sum_{t=1}^{t'} A_t^1 - \overline{V'}_{t'}^1, \underline{V'}_{t'}^2 - V_0^2 - \sum_{t=1}^{t'} A_t^2)$$

$$\alpha_{t'} = \min(V_0^1 + \sum_{t=1}^{t'} A_t^1 - \underline{V'}_{t'}^1, \overline{V'}_{t'}^2 - V_0^2 - \sum_{t=1}^{t'} A_t^2)$$

As such, only two constraints will be needed per time period, rather than four.

Rewriting the objective function. It is possible to reformulate the objective function as follows:

$$\max \sum_{t=1}^{T} \sum_{i=1}^{N} (\Lambda_t P_i - \Phi^1 D_i + \Phi^2 D_i) \cdot x_{t,i} + \Phi^1 \sum_{t=1}^{T} A_t^1 + \Phi^2 \sum_{t=1}^{T} A_t^2$$

Clearly, the external intakes being constants, their value do not change the optimal solution and we can discard them. Besides, we introduce $\Psi_{t,i} = \Lambda_t P_i - \Phi^1 D_i + \Phi^2 D_i$, being the value of operating point i at time period t. Note that the value of an operating point can be negative when Φ^1 is large and both Λ_t and Φ^2 are small. In the following we will drop the constant term from the objective function without loss of generality.

Rewriting the model. By using bounds $\beta_{t'}$ and $\alpha_{t'}$, and values $\Psi_{t,i}$, we obtain model defined as follows:

$$\max \sum_{t=1}^{T} \sum_{i=1}^{N} \Psi_{t,i} \cdot x_{t,i} \tag{3.2.5}$$

$$\text{s.t.} \quad \sum_{t=1}^{t'} \sum_{i=1}^{N} D_i x_{t,i} \geq \beta_{t'} \quad \forall t' \leq T \tag{3.2.6}$$

$$\sum_{t=1}^{t'} \sum_{i=1}^{N} D_i x_{t,i} \leq \alpha_{t'} \quad \forall t' \leq T \tag{3.2.7}$$

$$(3.1.6) \tag{3.2.8}$$

$$(3.1.9) - (3.1.15) \tag{3.2.9}$$

With this model, the objective function (3.2.5) is to maximize the value of each active operating point. Also, one can see (3.2.6) and (3.2.7) as resource windows, or (3.2.6) as nested covering constraints and (3.2.7) as nested knapsack constraints. However, instead of being resource windows on the volume, these constraints apply on the sum of the flows since time period 1. We denote such sum as the *cumulated flow*. For generality purposes, we define the cumulated flow between two time periods, rather than from the first time period.

Definition 1 (*Cumulated flow $\mathcal{D}_{t',t}$*)

The cumulated flow $\mathcal{D}_{t',t}$ is the sum of the flows from time periods t' to t:

$$\mathcal{D}_{t',t} = \sum_{t''=t'}^{t} \sum_{i=1}^{M} D_i x_{t'',i}$$

3.2.3 Bound tightening

The bounds β_t and α_t in (3.2.6) and (3.2.7) are obtained from the reservoir volumes, which can be very large compared to the water flow. Hence, an improvement of the model is to tighten these bounds.

At each time period, the flow is between 0 and $\sum_{i=1}^{M} D_i$. For any pair of time periods (t', t), with $t' < t$, the lower bound $\underline{\mathcal{D}}_{t',t} = 0$ and the upper bound $\overline{\mathcal{D}}_{t',t} = (t - t' + 1)\sum_{i=1}^{M} D_i$ are valid for the cumulated flow $\mathcal{D}_{t',t}$. We can therefore introduce bounds $\hat{\alpha}_t$ and $\hat{\beta}_t$ in the following way:

$$\hat{\alpha}_t = \min(\alpha_t, \overline{\mathcal{D}}_{1,t})$$
$$\hat{\beta}_t = \max(\beta_t, \underline{\mathcal{D}}_{1,t})$$

Bounds $\hat{\beta}_t$ and $\hat{\alpha}_t$ may be drastically tightened compared to β_t and α_t, especially for the first time periods. However, there is still room for tighter bounds. Suppose that at time period t the bounds are such that $\hat{\beta}_t > \hat{\beta}_{t+1}$. As the water flows are all non-negative, a solution verifying $\hat{\beta}_t$ at time period t cannot violate bound $\hat{\beta}_{t+1}$ at time period $t+1$. Hence, the latter can be tightened. Similarly, if $\hat{\alpha}_t + \overline{\mathcal{D}}_{t+1,t+1} < \hat{\alpha}_{t+1}$, a solution verifying $\hat{\alpha}_t$ at time period t cannot violate bound $\hat{\alpha}_{t+1}$ at time period $t+1$. This logic can be extended in order to tighten the bounds of any time period from the bounds of any other time period, following the rules below. Let a pair of time periods (t', t) with $t' < t$. Then, $\mathcal{D}_{1,t}$ lies in $[\hat{\beta}_{t'} + \underline{\mathcal{D}}_{t'+1,t}; \hat{\alpha}_{t'} + \overline{\mathcal{D}}_{t'+1,t}]$ and $\mathcal{D}_{1,t'}$ lies in $[\hat{\beta}_t - \overline{\mathcal{D}}_{t'+1,t}; \hat{\alpha}_t - \underline{\mathcal{D}}_{t'+1,t}]$.

Let us define $\tilde{\alpha}_t$ and $\tilde{\beta}_t$ as follows:

$$\tilde{\alpha}_t = \min(\min_{t'<t}(\hat{\alpha}_{t'} + \overline{\mathcal{D}}_{t'+1,t}), \min_{t'>t}(\hat{\alpha}_{t'} - \underline{\mathcal{D}}_{t+1,t'}))$$
$$\tilde{\beta}_t = \max(\max_{t'<t}(\hat{\beta}_{t'} + \underline{\mathcal{D}}_{t'+1,t}), \max_{t'>t}(\hat{\beta}_{t'} - \overline{\mathcal{D}}_{t+1,t'})$$

Tighter bounds α_t^* and β_t^* are calculated as follows:

$$\alpha_t^* = \min(\hat{\alpha}_t, \tilde{\alpha}_t)$$
$$\beta_t^* = \max(\hat{\beta}_t, \tilde{\beta}_t)$$

Note that computing all bounds β_t^* is of complexity T^2. Indeed, for a given t, computing β_t as well as $\tilde{\beta}_t$ both require one comparison, computing $\tilde{\beta}_t$ needs T comparisons and computing β_t^* requires one comparison. We obtain a similar complexity for computing upper bounds α_t^*.

Example 3

*Let us define an instance of the 1-HUC problem with $T = 6$. Bounds are $\beta_3 = 2$, $\beta_6 = 5$, $\alpha_3 = 2$, $\alpha_6 = 5$, and $\beta_t = -2$, $\alpha_t = 10$ for t in $\{1, 2, 4, 5\}$. The maximum flow is 2, and as we consider the 1-HUC without pump, the minimum flow is 0. By applying tighter bounds we can see that we drastically reduce the possibilities. In **Table 3.1a** the invalid values for the total flow at each time period, with respect to bounds β_t, are marked with a cross. **Table 3.1b** is similar to **Table 3.1a** with tighter bounds $\hat{\alpha}_t$ and $\hat{\beta}_t$, the crosses being in bold to emphasize the tightening of the bounds. **Table 3.1c** follows the same representation with the tightest bounds β_t^* and α_t^*.*

Table 3.1: Reducing the search space using bounds on the flows

(a) Table with bounds α_t and β_t

t	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
0															
1	X														X
2	X														X
3	X	X	X	X	X		X	X	X	X	X	X	X	X	X
4	X														X
5	X														X
6	X	X	X	X	X	X	X	X		X	X	X	X	X	X

(b) Table with bounds $\hat{\alpha}_t$ and $\hat{\beta}_t$

t	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
0															
1	X	**X**	**X**				**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	X
2	X	**X**	**X**						**X**	**X**	**X**	**X**	**X**	**X**	X
3	X	X	X	X	X		X	X	X	X	X	X	X	X	X
4	X	**X**	**X**										**X**	**X**	X
5	X	**X**	**X**												X
6	X	X	X	X	X	X	X	X		X	X	X	X	X	X

(c) Table with bounds α^*_t and β^*_t

t	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
0															
1	X	X	X				X	X	X	X	X	X	X	X	X
2	X	X	X				**X**	**X**	X	X	X	X	X	X	X
3	X	X	X	X	X		X	X	X	X	X	X	X	X	X
4	X	X	X	**X**	**X**				**X**	**X**	**X**	**X**	X	X	X
5	X	X	X	**X**	**X**	**X**				**X**	**X**	**X**	**X**	**X**	X
6	X	X	X	X	X	X	X	X		X	X	X	X	X	X

3.3 Resulting discretized 1-Hydro Unit Commitment problems and models for the remainder of the thesis

In this section, we define the discretized 1-Hydro Unit Commitment problem denoted by 1-HUC^{DRM}, that will be considered in the following chapters of this thesis. We define the corresponding model, resulting from the introduction of hydraulic constraints into M_{op} as presented in **Section 3.1** and the improvements proposed in **Section 3.2**. We also define a special case of the discretized 1-Hydro Unit Commitment problem without ramping nor min-up/down constraint, denoted by 1-HUC^{D}, and the associated model.

3.3.1 The discretized 1-Hydro Unit Commitment problems

The discretized 1-Hydro Unit Commitment problem (1-HUC^{DRM} problem). Consider a valley, with a plant between an upstream and a downstream reservoir, for which the fixed-head case is considered. The plant features N operating points, operating point i being a pair between a flow D_i and an associated power P_i. The time is discretized into T time periods. The ramping constraints limits the water flow increase from time period t to $t+1$ by an amount R_u (resp. R_d). The min-up constraints indicate that if operating point i is reached, it must remain active during L time periods. Similarly, the min-down constraints indicate that if operating point i is shut down, it cannot be reached during L time periods. For each time period $t \leq T$, there is an upper bound α_t^* and a lower bound β_t^* of the cumulated flow $\mathcal{D}_{1,t}$. For each time period $t \leq T$, each operating point $i \leq N$ has a value $\Psi_{t,i}$. Solving the 1-HUC^{DRM} problem is to schedule the operating points such that all constraints are satisfied, and the value is maximized.

The 1-HUC^{D} problem. We define the 1-HUC^{D} problem as the simplified case without ramping constraints and min-up/down constraints.

3.3.2 Mathematical models

Model $M_{op\text{-}DRM}$. Let $x_{t,i}$ be the binary variable indicating that operating point i is active at time period t. The model $M_{op\text{-}DRM}$ is as follows.

$$\max \sum_{t=1}^{T}\sum_{i=1}^{N} \Psi_{t,i} \cdot x_{t,i} \tag{3.2.5}$$

$$\text{s.t.} \quad \sum_{t=1}^{t'}\sum_{i=1}^{N} D_i x_{t,i} \geq \beta_{t'}^*, \qquad \forall t' \leq T \tag{3.2.6}$$

$$\sum_{t=1}^{t'}\sum_{i=1}^{N} D_i x_{t,i} \leq \alpha_{t'}^*, \qquad \forall t' \leq T \tag{3.2.7}$$

$$x_{t,i} \geq x_{t,i+1}, \qquad \forall t \leq T, \forall i \leq N-1 \tag{3.1.6}$$

$$\sum_{i\in\mathcal{N}} D_i x_{t,i} - \sum_{i\in\mathcal{N}} D_i x_{t-1,i} \leq R_u, \qquad \forall t \in \{2,\dots,T\} \tag{3.1.9}$$

$$\sum_{i\in\mathcal{N}} D_i x_{t-1,i} - \sum_{i\in\mathcal{N}} D_i x_{t,i} \leq R_d, \qquad \forall t \in \{2,\dots,T\} \tag{3.1.10}$$

$$\sum_{t'=t-L+1}^{t} v_{t',i} \leq x_{t,i} \qquad \forall t \in \{L,\dots,T\}, \forall i \in \mathcal{N} \tag{3.1.11}$$

$$\sum_{t'=t-L+1}^{t} v_{t',i} \leq 1 - x_{t-L,i} \qquad \forall t \in \{L,\dots,T\}, \forall i \in \mathcal{N} \tag{3.1.12}$$

$$v_{t,i} \geq x_{t,i} - x_{t-1,i} \qquad \forall t \in \{2,\dots,T\}, \forall i \in \mathcal{N} \tag{3.1.13}$$

$$x_{t,i} \in \{0,1\}, \qquad \forall t \leq T, \forall i \in \mathcal{N} \tag{3.1.14}$$

$$v_{t,i} \in \{0,1\}, \qquad \forall 2 \leq t \leq T, \forall i \in \mathcal{N} \tag{3.1.15}$$

Model $M_{op\text{-}D}$. We define model $M_{op\text{-}D}$ similar to $M_{op\text{-}DRM}$, without ramping nor min-up/down constraints. The resulting model is (3.2.5)-(3.2.7), (3.1.6) and (3.1.14).

3.4 Conclusion

In this chapter we define the model retained for the study of hydraulic constraints in the remainder of the thesis. These constraints are mainly resource windows, min-up/down and ramping constraints. As the head does not play a major role for these particular constraints, we focus on the fixed-head 1-HUC problem. Between the three most efficient models highlighted for the fixed-head case in **Chapter 2**, model M_{op} provides the best precision. Besides, M_{op} is linear in the fixed-head case, and in this case it is possible to consider many more operating points than in the comparison. These additional operating points can reduce some of the drawbacks observed in the modeling comparison. Hence, we introduced a generalization of M_{op}, taking into account pumps as well as ramping and min-up/down constraints. We reformulated the model so that bounds on the volume become nested knapsack and covering inequalities, which can also be defined as resource windows for the cumulated flow. Moreover, we improved this model using a bound-tightening technique, which can be done in polynomial time. Finally, we defined the resulting specific 1-Hydro Unit Commitment problem variants considered for the remainder of the thesis, denoted 1-HUC^{DRM} and 1-HUC^{D}, as well as the corresponding models denoted $M_{op\text{-}DRM}$ and $M_{op\text{-}D}$.

In the next chapter, we will conduct a polyhedral study on a variant of the Knapsack Problem, designed to be the combinatorial core of the 1-HUC^{D} problem. From that study, we will also propose a dedicated two-phase Branch& Cut algorithm to solve the knapsack problem variant.

Part II

Polyhedral studies

4 Polyhedral study of the combinatorial core of the discretized 1-Hydro Unit Commitment problem: the Symmetric-weight Chain Precedence Knapsack Problem

Table of contents

In the previous chapter, we presented a comparison of modeling alternatives, from which we selected a linear model. Such a model is similar to the one considered at EDF to provide solutions to the 1-Hydro Unit Commitment problem. As such, conducting a polyhedral study on this linear model could improve the current approach at EDF. The idea is to focus on the combinatorial aspects, which means considering the relationship between the upperbound on cumulated flow and the discrete set of flows. For this purpose, we define a variant of the knapsack problem, with Symmetric weight and Chain Precedences (SCPKP).

In this chapter, we carry out a polyhedral study of the Symmetric-weight Chain Precedence Knapsack Problem. Then, we present a two-phase Branch & Cut algorithm, exploiting the symmetric aspect of the problem through a structure called pattern. In **Section 4.1** we define the SCPKP and its relationship to the 1-HUC problem. In **Section 4.2** we review polyhedral results on related knapsack variants. In **Section 4.3** we present the complexity of the considered problem and first polyhedral results. In **Section 4.4** we define the patterns and their corresponding set of valid inequalities, which are also facet-defining in some cases. In **Section 4.5** we present the separation problem for this new set of valid inequalities. In **Section 4.6** we detail the algorithms featured in our Branch & Cut scheme. In **Section 4.7** we compare our Branch & Cut scheme to two state-of-the-art Branch & Cut variants. In **Section 4.8**, we draw concluding remarks.

4.1 Definition of the Symmetric-weight Chain Precedence Knapsack Problem

The Symmetric-weight Chain Precedence Knapsack Problem (SCPKP) is defined as follows. Consider I groups of J elements, where I and J are positive integers. Let item (i, j) be element j of group i. Item (i, j) has weight $W_j \in \mathbb{R}_{\geq 0}$ and value $V_{ij} \in \mathbb{R}$. This means that the weight of item (i, j) does not depend on the group index i. Within each group, order constraints are such that any item (i, j) can be selected only if item $(i, j-1)$ is selected, thus inducing chain precedence constraints. Let C be the maximum knapsack capacity. Solving the SCPKP is to maximize the total value of the selected items, while the chain precedence constraints are satisfied, and the total weight of the selected items is less than or equal to capacity C. **Figure 4.1** depicts the SCPKP, where a square represents an item, and the arrows represent the chain precedence constraints.

We define an instance of the SCPKP as follows.

Definition 2 (*Instance of the SCPKP*)

We denote $(I, J, W, V\ C)$ an instance of the SCPKP, with I the number of groups, J the number of

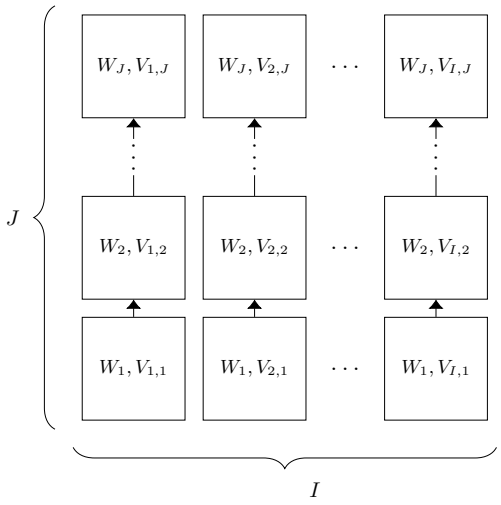


Figure 4.1: Graphical representation of the SCPKP

elements, W the weight vector, V the value matrix and C the knapsack capacity.

Studying the SCPKP is motivated by the fact that its constraints are a subset of the ones for the 1-HUC^D problem. More precisely, the constraints of the SCPKP are similar to the ones of the 1-HUC^D problem considering only one of the knapsack inequalities (3.2.7). This is beneficial as the polyhedron of the SCPKP is easier to study and results are extendable to the 1-HUC^D problem.

4.2 Related knapsack polytopes

The Knapsack Problem (KP) and its variants have been widely studied in the literature [53]. The SCPKP is a knapsack variant which has not been studied yet. Nevertheless, it can be related to some variants of the knapsack problem in the literature. As the chain precedence constraints are a special case of precedence constraints, the CPKP and SCPKP are special cases of the Precedence Knapsack Problem (PKP) [17]. An alternative model for the CPKP and SCPKP is to consider disjunctive constraints rather than chain precedence constraints. Hence, the CPKP and SCPKP are special cases of the Multiple-choice Knapsack Problem (MCKP) [58] where groups are not restricted to have the same number of items. The latter is also a special case of the Disjunctive Knapsack Problem (DKP) [92]. Both the PKP and the DKP are generalizations of the classical Binary Knapsack Problem (BKP) [7]. Indeed, the BKP is a PKP without any precedence, as well as a DKP without any disjunctive constraints. Interestingly, the CPKP is also a generalization of the BKP, whereas the SCPKP is not. Indeed, an CPKP with exactly one item for each group does not have any chain precedence constraints, which is exactly a BKP. However, because of the symmetric weight, an SCPKP with one item per group would be a trivial BKP, where each item has the exact same weight. Furthermore, the SCPKP is not a Multiple Knapsack Problem (MKP) [39] despite the presence of multiple groups. Indeed, a capacity constraint is applied on all groups simultaneously

in the SCPKP, whereas it is applied for each group individually in the MKP. As the chain precedence are a special case of the precedence constraints, it is worth mentioning the Multi-Period PKP (MPPKP) [75]. This is a generalization of the PKP, where items can be selected at different time periods. This adds another set of precedence constraints, as if an item is selected at a given time period, then it is also selected for the following time periods. In the case of an MPPKP with chain precedence constraints, its representation is identical to the SCPKP in **Figure 4.1** but with additional arcs going from left to right. Note that in such a case, the MPPKP is an SCPKP where all the symmetries with respect to the groups are broken. However, to the best of our knowledge, there is no study for such special case of the MPPKP.

In this section, we present ILP formulations and valid inequalities of related knapsack problems. We focus on generalizations of the SCPKP, namely the PKP and the DKP. As facet-defining inequalities for the BKP have been extended to variants related to the SCPKP, the BKP is also considered in this short state-of-the-art review of polyhedral results of interest for the SCPKP.

4.2.1 Binary knapsack polytope

The SCPKP is neither a generalization, nor a special case of both the BIKP [20] and the BKP [7]. Hence, the facet-defining inequalities of these problems may not be related to those of the SCPKP. However, we introduce inequalities of the BKP as they have been extended to variants that are related to the SCPKP.

Let V be a set of items, the BKP can be formulated as follows:

$$\begin{aligned} \max & \sum_{j \in V} v_j x_j \\ \text{s.t.} & \sum_{j \in V} w_j x_j \leq C, \\ & x_j \in \{0,1\} \qquad\qquad \forall j \in V. \end{aligned}$$

In [7] are defined *covers*, *minimal covers* and *minimal cover inequalities* as follows. A cover $U \subseteq V$ is a set of items such that $\sum_{j \in U} w_j > C$. A minimal cover $U \subseteq V$ is a cover such that no proper subset of U is also a cover, i.e., $\sum_{j \in U \setminus \{i\}} w_j \leq C$, for each $i \in U$. Let U be a minimal cover. The minimal cover inequality associated with U is:

$$\sum_{j \in U} x_j \leq |U| - 1.$$

The family of minimal cover inequalities contains facet-defining inequalities of the BKP.

Other inequalities for the BKP have been defined, namely the extended cover inequalities, the $(1, k)$-configurations inequalities and the weight inequalities [30]. Moreover, the authors prove that for each of these three families of inequalities, the separation problem is NP-hard. More recently, a new lifting for minimal cover inequalities is introduced in [65]. As the aforementioned inequalities have not been extended to the PKP, and therefore may not extend to the SCPKP, we do not detail them further.

4.2.2 Disjunctive constrained Knapsack Problem

As previously mentioned, the SCPKP is a special case of the MCKP. However, to the best of our knowledge, there are no polyhedral results related to the MCKP. As such, we review a more generic problem, the DKP formulated as follows. Consider a disjunctive graph defined by vertex set V corresponding to items and by edge set E corresponding to conflicts between items. A compact multiple choice formulation of the DKP is:

$$\begin{aligned}
&\max \sum_{j \in V} v_j x_j \\
&\text{s.t.} \sum_{j \in V} w_j x_j \leq C, \\
&x_j + x_i \leq 1 && \text{if } (j,i) \in E, \\
&x_j \in \{0,1\} && \forall j \in V.
\end{aligned}$$

Five families of inequalities containing facet-defining inequalities have been reported for the DKP [92]: the clique inequalities, the cover inequalities, odd-cycle and hypergraph inequalities, the clique-cover inequalities, the clique-cover-partition inequalities. The cover inequality for a set $U \subseteq V$ is close to the one of the BKP where U is such that there is no edge between two items of U in the disjunctive graph. Extensions of cover inequalities such as clique-cover inequalities and clique-cover-partition inequalities also rely on cliques in the disjunctive graph. The odd-cycle inequalities and their extension, the hypergraph inequalities, apply to cycles in the disjunctive graph. As for the precedence graph of the SCPKP, the corresponding disjunctive graph is very special. Indeed, in the precedence graph of the SCPKP as depicted in **Figure 1.6**, arcs representing chain precedence constraints are exclusively between elements in the same group. The SCPKP can be translated into a disjunctive form: where a single element can be selected in each group. The associated disjunctive graph is such that the vertices representing disjunctive constraints are also exclusively between elements in the same group. Consequently, every group is a clique, and no item is adjacent to an item in another group. Because of this specific disjunctive graph, the clique inequalities cannot be adapted to the SCPKP, and similarly for the odd-cycle and hypergraph inequalities. As the clique-cover and clique-cover-partition inequalities rely on cliques, these inequalities cannot be adapted to the SCPKP either.

4.2.3 Precedence constrained knapsack polytope

As stated previously, the SCPKP is a special case of the PKP [17], defined as follows. Let $(V, \preceq)$ be a partially ordered set of items. Item j covers item i if there is no k such that $j \prec k \prec i$. A formulation of the PKP is:

$$\max \sum_{j \in V} v_j x_j$$

$$\begin{aligned} &\text{s.t.} \sum_{j \in V} w_j x_j \leq C, \\ &x_j \leq x_i && \text{if } j \text{ covers } i, \\ &x_j \in \{0,1\} && \forall j \in V. \end{aligned}$$

The minimal cover and related inequalities have been extended to the PKP [17]. To do so, *lower-ideals* are defined as sets $U \subset V$ such that if $i \in U$ and $j \prec i$, then $j \in U$. The minimal cover inequalities have been extended to the case of the PKP with U a minimal cover as well as a lower-ideal:

$$\sum_{j \in U} x_j \leq |U| - 1.$$

The minimal cover inequalities have been enhanced in the literature with various liftings. In [17] a lifting procedure is described, starting from a facet of a lower-dimensional polyhedron. The lifting procedure provides valid inequalities that are not necessarily facet-defining for the PKP.

The minimal cover can also be adapted to the PKP [80] [64] without the use of lower-ideals. Let A be the arcs of the precedence graph. The arc set A is modified such that if $(i,j) \in A$ and $(j,k) \in A$ then $(i,k) \in A$. Let U be a variable set. We define U_p the *set of predecessors* of U. More precisely, U_p contains all $i \notin U$ such that $(i,j) \in A$ and $j \in U$. An *induced cover* is a set $U \subseteq V$, such that there is no arc between a pair of items in U and $\sum_{j \in U} w_j + \sum_{k \in U_p} w_k \geq C$. A *Minimal Induced Cover* (MIC) is an induced cover U such that $\sum_{j \in U \setminus \{i\}} w_j + \sum_{k \in U_p} w_k \geq C$, for each $i \in U$. The *MIC inequalities* are as follows, with U a MIC:

$$\sum_{j \in U} x_j \leq |U| - 1.$$

These MIC inequalities have been enhanced with a lifting procedure [80]. From a MIC U, the procedure described uses items that have at least two successors of the precedence graph in U. A sequential lifting procedure has also been developed [64]. Consider U a MIC and U_p the predecessors of U, we define $U_r = V \setminus (U \cup U_p)$. The sequential lifting procedure computes the coefficients α_j (resp. β_j), for each $j \in U_p$ (resp. $j \in U_r$), leading to a lifted inequality:

$$\sum_{j \in U} x_j + \sum_{j \in U_p} \alpha_j (1 - x_j) + \sum_{j \in U_r} \beta_j x_j \leq |U| - 1.$$

Computing the optimal values of coefficients α_j, defined as *downlifting*, can be done in polynomial time. However computing the optimal values of coefficients β_j, defined as *uplifting*, is proven to be NP-hard.

In [37] is defined a separation procedure, which computes bounds α_j and β_j in a reduced precedence graph. The results show that computing bounds β_j takes large computational times, even if the problem is relaxed. This procedure is further detailed in the following **Section 4.2.4**.

In [15] is defined the clique-based inequalities for the PKP. For this purpose, a disjunctive graph is introduced, where a vertex corresponds to an item. The edges are between each pair of vertices corresponding to a pair of items that cannot be into a feasible solution simultaneously without violating the capacity constraint. Such a type of inequality does not seem to adapt efficiently to the case of the SCPKP.

Indeed, the disjunctive graph described does not have any edge if $C \geq 2\sum_{j=1}^{J} w_j$, which is the case of the majority of large size instances of the SCPKP.

4.2.4 Specialization of the Minimal Induced Cover inequalities to the chain precedence constraints of the Symmetric-weight Chain Precedence Knapsack Problem.

As seen previously, only the minimal cover inequalities [17] and (lifted) MIC inequalities [80][64] defined for the PKP are likely to be facet-defining for the SCPKP. From the inequalities of the PKP previously described, clique-based inequalities [15] do not apply to the SCPKP, and downlifted and uplifted MIC inequalities [64] dominate minimal cover inequalities [17] and MIC inequalities [80].

Downlifted and uplifted MIC inequalities can be adapted to the SCPKP, by introducing α_{ij} and β_{ij}. In this case, they have the following form:

$$\sum_{(i,j)\in U} x_{ij} + \sum_{(i,j)\in U_p} \alpha_{ij}(1 - x_{ij}) + \sum_{(i,j)\in U_r} \beta_{ij} x_{ij} \leq |U| - 1;$$

where U is a MIC, U_p the set of predecessors of U and U_r the set of all items that are not in $U \cup U_p$. First, we have the following property.

Property 1

Let U be a MIC for the SCPKP, for all $(i,j) \in U_p$, the downlifting procedure yields $\alpha_{ij} = 0$.

The proof and an illustration are given in B.1.1. This property means that there is no need to downlift an inequality for the SCPKP, as it can only yield coefficient 0. However, there is no similar proof for coefficients β_{ij}, i.e., for the uplifting. Hence, we introduce the Uplifted MIC (UMIC) inequalities for the SCPKP, with the following form.

Definition 3 (*UMIC inequalities for the SCPKP*)

Let U be a MIC, U_p be the set of predecessors of U and U_r be the set of all items that are not in $U \cup U_p$. The UMIC inequalities applied for the SCPKP have the following form:

$$\sum_{(i,j)\in U} x_{ij} + \sum_{(i,j)\in U_r} \beta_{ij} x_{ij} \leq |U| - 1.$$

Valid UMIC inequalities can be obtained using the procedure of [37], and are experimentally compared to the inequalities introduced in this chapter in **Section 4.7**.

4.3 Complexity and polytope

In this section, we present preparatory results for the SCPKP. We first show that the problem is NP-hard. Then, we define a formulation for which we present preliminary polyhedral properties.

4.3.1 Complexity

Before stating the complexity of the SCPKP, we identify two special cases where the SCPKP is easy to solve.

Property 2

Let (I, J, W, V, C) be an instance of the SCPKP. If $I = 1$ or $J = 1$, the SCPKP can be solved in polynomial time.

Proof : When $J = 1$, there is a single group. Due to chain precedence constraints, there are only I feasible solutions. One can enumerate all solutions in polynomial time.

When $I = 1$, then there is a single item per group. As the weight W_j is similar for all group, one can add items by decreasing value V_{i1} until the capacity C is reached, or until there is no group with $V_{i1} > 0$. ■

In the following, we then only consider instances of the SCPKP with $I \geq 2$ and $J \geq 2$.

To state the complexity of the SCPKP, we recall the following two problems of the literature. The first problem is the Unbounded Integer Knapsack Problem (UIKP), which is a knapsack problem where items can be selected an unlimited number of times. The second problem is the Subset Sum Problem (SSP), where considering a set of positive integers S, and a target integer C', the aim is to find $S' \subseteq S$ such that the sum of all elements of S' equals C'. First a reduction from the UIKP allows us to prove the NP-hardness of the SCPKP when $I \geq \frac{C}{W_1}$. Then a reduction from the (SSP) enables us to prove the NP-hardness of the SCPKP when $I \geq J$.

Theorem 1

The SCPKP is NP-hard when $I \geq \frac{C}{W_1}$

Proof : Let (J', W', V', C') be an instance of the UIKP with J' items, item $j \leq J'$ has weight W'_j and value V'_j. We assume without loss of generality that $W'_j > W'_{j-1}$. The aim is to maximize the value of the selected items, each can be selected multiple times, without any limitation, while satisfying the maximum knapsack capacity C'.

From (J', W', V', C'), one can construct instance (I, J, W, V, C) of the SCPKP with same value. The number of elements is $J = J'$, the number of groups is $I = \lceil \frac{C'}{\min(W'_j)} \rceil$. The weights are $W_1 = W'_1$ and $W_j = W'_j - W'_{j-1}$ for each $j > 1$. The values are $V_{i,1} = V'_1$ with $i \in \{1,\dots,I\}$ and $V_{i,j} = V'_j - V'_{j-1}$ for each $j \in \{2,\dots,J\}$ and $i \in \{1,\dots,I\}$. Finally, the capacity is $C = C'$. The aim is to maximize the value of the selected items, verifying the chain precedence constraints, and satisfying the maximum knapsack capacity C.

For a solution of (J', W', V', C) there is a corresponding solution of (I, J, W, V, C) with same value. Let p_j be the number of times item j is selected in a solution of (J', W', V', C). There is a solution of (I, J, W, V, C) that is equivalent: for each element j, there are p_j unique groups of (J', W', V', C) where exactly elements 1 to j are selected. Such a solution of (I, J, W, V, C) exists as $I = \lceil \frac{C}{\min(W'_j)} \rceil$, then $\sum_{j=1}^{n} p_j \leq I$.

Similarly, for a solution of (I, J, W, V, C) there is a solution of (J', W', V', C) with same value. Let p_j be the number of groups where exactly elements 1 to j are selected in a solution of (I, J, W, V, C). There is a solution of (J', W', V', C) that is equivalent: each item j is selected exactly p_j times.

Thus, the UIKP is a special case of the SCPKP, and because the UIKP is NP-hard [111], the SCPKP is NP-hard when $I \geq \frac{C}{W_1}$. ■

Theorem 2

The SCPKP is NP-hard when $I \geq J$

Proof : Let (S, C') be an instance of the (SSP), with $S = \{W'_1, \ldots, W'_{J'}\}$, $a_j > 0$ for all $j \leq J'$, and $C' > 0$. Without loss of generality, we suppose $W'_j \leq W'_{j+1}$ for all $j < J'$. The aim is to find $S' \subseteq S$ such that $\sum_{i \in S'} W'_i = C'$.

From (S, C'), one can construct instance (I, J, W, V, C) of the SCPKP as follows. The number of element $J = |S|$, the number of groups I can have any value greater than or equal to J. For the purpose of this proof, we consider $I = J$. Weights are $W_1 = W'_1$, $W_j = W'_j - W'_{j-1}$ for all $j > 1$. Values are $V_{ij} = 0$ if $j \neq i$ and $V_{ij} = W'_j$ if $i = j$. Finally, capacity is $C = C'$. We consider here the decision problem associated with the SCPKP, where the aim is to find a solution of value greater or equals to C.

By construction of the weights and values, for a given group $i \leq I$, $\sum_{j=1}^{i} W_j = \sum_{j=1}^{i} V_{ij} = W'_j$, and for any $j' > 0$, $j' \neq i$, $\sum_{j=1}^{j'} W_j > \sum_{j=1}^{j'} V_{ij}$. The decision problem of the SCPKP indicates that the solution must have value above or equal to C, and weight below or equal to C. Consequently, a feasible solution can only include, for a group i, exactly the i first elements, or no element.

For a solution of (I, J, W, V, C), there is a corresponding solution of (S, C'). One can consider S' with all values W'_i if elements are selected in group i. Besides, solution of (I, J, W, V, C) has its weight equal to its value, hence they must both be equal to $C = C'$. Consequently, $\sum_{i \in S'} W'_i = C'$.

For a solution of (S, C'), there is a corresponding solution of (I, J, W, V, C). Let S' be the solution of the (SSP). For each $i \in S'$, select exactly the first i elements of group i in the SCPKP. By construction, this solution of the SCPKP is valid, as it has both weight and value equal to $\sum_{i \in S'} W'_i = C' = C$.

As the (SSP) is NP-hard [44], then the SCPKP is NP-hard when $I \geq J$. ■

Note that the case of the SCPKP where $I < \min(\frac{C}{W_1}, J)$ remains an open question.

The SCPKP, and the CPKP, are generalizations of the Unbounded Integer Knapsack problem UIKP. Consequently, the CPKP is a generalization of both the BKP and the UIKP. The Bounded Integer Knapsack problem (BIKP) [20] is also a generalization of the BKP and the UIKP. Indeed, the BKP is a special case with upper bounds 1 on the number of repetition for each item, and the UIKP is a special case with finite upper bound sufficiently large to not be restrictive. However, the BIKP and the SCPKP are not related. In fact, the BIKP considers a maximum number of repetitions for each item, while the SCPKP

would be closer to an Integer Knapsack problem with a shared upper bound on the total number of items, repetition included. This shared upper bound is the number of groups I. As the chain precedence constraints are specific precedence constraints, the CPKP is also a special case of the Precedence Knapsack problem PKP. The chain precedence constraints can also be modeled as disjunctive constraints, hence the CPKP is also a special case of the Disjunctive Knapsack Problem DKP.

All connections between the knapsack problem variants are depicted in **Figure 4.2** with a graph in which each vertex represents a variant and each arc indicates that the variant at the tail is a generalization of the variant at the head.

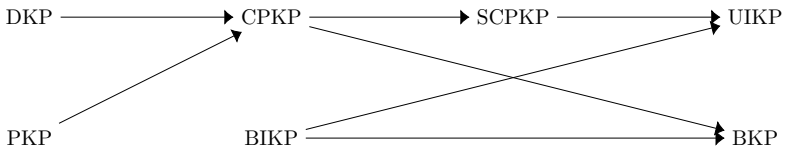


Figure 4.2: Generalization orders of some variants of the knapsack problem

4.3.2 Formulation

Let x_{ij} be a binary variable such that $x_{ij} = 1$ if item (i, j) is selected in the solution. We denote $\mathcal{V}$ the set of variables x_{ij} for the SCPKP. The total number of variables is $n = I \times J$. The SCPKP can be formulated with the Integer Linear Program (ILP) M_{SCPK} as follows.

$$\max_{x_{ij}\in\{0,1\}} \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij}x_{ij}$$

$$\text{s.t.} \sum_{i=1}^{I}\sum_{j=1}^{J} W_j x_{ij} \leq C, \tag{4.3.1}$$

$$x_{ij} \leq x_{ij-1} \qquad \forall x_{ij} \in \mathcal{V}, j \geq 2, \tag{4.3.2}$$

$$x_{ij} \geq 0 \qquad \forall x_{ij} \in \mathcal{V},$$

$$x_{ij} \leq 1 \qquad \forall x_{ij} \in \mathcal{V}.$$

In this formulation, the objective function is to maximize the total value of the selected items. Inequality (4.3.1) is the capacity constraint, inequalities (4.3.2) correspond to the chain precedence constraints. We define the polytope P_{SCPKP} as the convex hull of the feasible solutions of the SCPKP:

$$P_{SCPKP} = \text{conv}\Big\{x \in \{0,1\}^n : x \text{ satisfies (4.3.1)} - \text{(4.3.2)}\Big\}.$$

The proposed formulation is the so-called *incremental formulation* [71]. It is also possible to define a *multiple choice* formulation, featuring disjunctive constraints instead of chain precedence constraints

(4.3.2). Both formulations yield the same LP relaxation [27], but we do not further detail the multiple choice formulation, as it is not necessary for the purpose of this chapter.

We can remark that the SCPKP features symmetries. Indeed, items have symmetric weights, the chain precedence constraints (4.3.2) are the same for each group, and the capacity constraint (4.3.1) applies to all groups. Hence, from a feasible solution, one can obtain another feasible solution by a permutation of the groups. However, the values of such solutions are not necessarily the same, as there are no symmetries in the values. We define *polyhedral symmetry* as a symmetry restricted to the constraints set.

Definition 4

A polyhedral symmetry is a permutation π of the variables such that for any feasible solution x, $\pi(x)$ is also a feasible solution.

In other words, a polyhedral symmetry only concerns the constraints. This is a generalization of the classical symmetry which also restricts the objective values of two symmetric solutions to be equal. In the case of the SCPKP, for any feasible solution, any permutation of groups yields another feasible solution, but their respective values are not necessarily equal. Classical methods to handle symmetries [69] may not apply to polyhedral symmetries. In particular, the idea is not to find a representative solution and discard the symmetric solutions, but rather to capture all polyhedral symmetric solutions in a special structure providing an efficient way to find an optimal solution.

4.3.3 First polyhedral properties

Definition 5 (*Full-dimensional condition (fd)*)

An SCPKP verifies (fd) if any item (i, j), can be selected in at least one feasible solution.

Property 3

Any instance of the SCPKP that does not verify (fd) can be transformed into an instance of the SCPKP that verifies (fd) , with the exact same solutions.

Proof : Let (I, J, W, V, C) be an instance of the SCPKP where an item (i, j) cannot be selected in any feasible solution, and $(i, j-1)$ can be selected in a feasible solution. Clearly, (I, J, W, V, C) does not verify (fd) . Because of the symmetric weights, item j cannot be selected in any group, therefore, for any $i \leq I$ and $j' \geq j$, $x_{ij'} = 0$ in any integer solution. It is possible to create another instance of the SCPKP $(I, j-1, W, V, C)$ with the exact same integer solutions. Because $(i, j-1)$ can be selected in a feasible solution, and because of the symmetric weight, any item of $(I, j-1, W, V, C)$ can be selected. Hence, $(I, j-1, W, V, C)$ verifies (fd) . ■

Without loss of generality, in the following we will only consider SCPKP instances verifying (fd) .

Definition 6 (*Solution $X_{\mathcal{X}}$*)

For a variable set $\mathcal{X}$, solution $X_{\mathcal{X}}$ is the solution with $x_{ij} = 1$ for each $(i,j) \in N \times M$ such that $\exists j' \geq j$ and $x_{ij'} \in \mathcal{X}$, and $x_{ij} = 0$ otherwise.

We define the special case X_{ij} if $\mathcal{X} = \{x_{ij}\}$, and $X_{\emptyset}$ if $\mathcal{X} = \emptyset$.

Example 4

Let $(3,3,W,V,C)$ be an instance of the SCPKP. Let $\mathcal{X} = \{x_{12}, x_{21}, x_{31}, x_{33}\}$ be a set of variables. The solution $X_{\mathcal{X}}$ is with $x_{11} = x_{12} = x_{21} = x_{31} = x_{32} = x_{33} = 1$ and $x_{13} = x_{22} = x_{23} = 0$.

Note that the solution $X_{\mathcal{X}'}$ associated to the set $\mathcal{X}' = \{x_{12}, x_{21}, x_{33}\}$ is the same as the solution $X_{\mathcal{X}}$.

To ensure that the chain precedence constraints are taken into account while considering a solution $X_{\mathcal{X}}$, we introduce the set weights associated with a set of variables $\mathcal{X}$.

Definition 7 (*Set weights*)

For a given set of variables $\mathcal{X}$, and for all $i \leq I$, $j \leq J$, the set weights are

$$s_{ij}(\mathcal{X}) = \begin{cases} 0 & \text{if } x_{ij} \notin \mathcal{X}, \\ \sum_{k=j'+1}^{j} W_k & \text{if } x_{ij} \in \mathcal{X} \text{ with } j' = \max\{j' | x_{ij'} \in \mathcal{X}, j' < j\}, \\ \sum_{k=1}^{j} W_k & \text{if } x_{ij} \in \mathcal{X} \text{ and } x_{ij'} \notin \mathcal{X}, \forall j' < j. \end{cases}$$

Example 5

Let $(3,3,[1,3,2],V,C)$ be an instance of the SCPKP. Let $\mathcal{X} = \{x_{12}, x_{21}, x_{31}, x_{33}\}$ be a set of variables. The coefficients are $s_{12}(\mathcal{X}) = W_1 + W_2 = 4$, $s_{21}(\mathcal{X}) = W_1 = 1$, $s_{31}(\mathcal{X}) = W_1 = 1$, $s_{33}(\mathcal{X}) = W_2 + W_3 = 5$ and 0 otherwise.

The coefficient $s_{ij}(\mathcal{X})$ embeds the chain precedence constraints. Indeed, if $x_{ij} = 1$, $x_{ij} \in \mathcal{X}$, then all $x_{ij'} = 1$, $j' \leq j$, even for $x_{ij'} \notin \mathcal{X}$. Thus, if $x_{ij} = 1$ then the weights of all variables $x_{ij'} \notin \mathcal{X}$ should be accounted for, which is the purpose of coefficients $s_{ij}(\mathcal{X})$.

For the sake of simplicity, we define $\mathcal{Y}$ as a k-intersection of $\mathcal{X}$ if $\mathcal{Y}$ contains k elements of $\mathcal{X}$.

Definition 8 (*k-intersection*)

Let $\mathcal{X}$, $\mathcal{Y}$ be sets of variables. Variable set $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$ if $|\mathcal{Y} \cap \mathcal{X}| = k$, for each variable $x_{ij} \in \mathcal{Y}$ if $x_{ij'} \in \mathcal{X}$ with $j' \leq j$, then $x_{ij'} \in \mathcal{Y}$ and $\sum_{x_{ij} \in \mathcal{Y}} s_{ij}(\mathcal{Y}) \leq C$.

If $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$, the reverse can also be true. Consequently, a k-intersection of $\mathcal{X}$ is not necessarily a subset of $\mathcal{X}$.

Property 4

Let $\mathcal{X}$ be a set of variables. Let $\mathcal{Y}$ be a k-intersection of $\mathcal{X}$. There exists a set $\mathcal{Y}' \subseteq \mathcal{Y}$, a k-intersection of $\mathcal{X}$ with $|\mathcal{Y}'| = k$.

Proof : Consider $\mathcal{Y}' = \mathcal{Y} \cap \mathcal{X}$. Clearly, $|\mathcal{Y}'| = k$ and $\mathcal{Y}' \cap \mathcal{X} = k$. Besides, $\mathcal{Y}' \subseteq \mathcal{Y}$, all weights are positive, and there are chain precedence constraints, hence if $X_{\mathcal{Y}}$ is valid, so is $X_{\mathcal{Y}'}$. Consequently, $\mathcal{Y}'$ is also a k-intersection of $\mathcal{X}$. ■

Example 6

Let $(3, 3, [1, 3, 2], V, 12)$ be an instance of the SCPKP. Let $\mathcal{X} = \{x_{11}, x_{13}, x_{22}, x_{31}\}$ and $\mathcal{Y} = \{x_{11}, x_{13}, x_{21}, x_{22}\}$ be two variable sets. In this case, $\mathcal{Y}$ is a 3-intersection of $\mathcal{X}$. Indeed, $|\mathcal{X} \cap \mathcal{Y}| = 3$, $s_{11}(\mathcal{Y}) + s_{13}(\mathcal{Y}) + s_{21}(\mathcal{Y}) + s_{22}(\mathcal{Y}) = 1+5+1+3 = 10 \leq C = 12$. However, $\mathcal{X}$ is not a 3-intersection of $\mathcal{Y}$. Indeed, $|\mathcal{X} \cap \mathcal{Y}| = 3$ and $s_{11}(\mathcal{X}) + s_{13}(\mathcal{X}) + s_{22}(\mathcal{X}) + s_{31}(\mathcal{X}) = 1 + 5 + 4 + 1 = 11 \leq C = 12$, but there is $x_{21} \in \mathcal{Y}$ and $x_{21} \notin \mathcal{X}$ even if $x_{22} \in \mathcal{X}$.

Theorem 3

P_{SCPKP} is full dimensional.

Proof : As there are I groups and J elements, solutions X_{ij} and $X_{\emptyset}$ yield a total of $I \times J + 1 = n + 1$ different solutions, which are feasible, otherwise (fd) is not verified. For any i, solution X_{iJ} is the only solution with $x_{iJ} = 1$, thus being affinely independent to other solutions. For any i, solution X_{ij}, $j < J$ is the only solution with $x_{ij} = 1$ and $x_{ij+1} = 0$, this being affinely independent to other solutions. Clearly, $X_{\emptyset}$ is affinely independent to other solutions, which means that there are $n+1$ affinely independent solutions. ■

With the dimension of P_{SCPKP} it becomes possible to characterize when chain precedence inequalities and trivial inequalities with bounds at 0 are facet-defining.

Theorem 4

Inequalities $x_{ij} \geq 0$ are facet-defining for P_{SCPKP} if and only if $j = J$.

Proof : As there are I groups and J elements, solutions X_{ij} and $X_{\emptyset}$ yield a total of $I \times J + 1 = n + 1$ different solutions, which are feasible otherwise (fd) is not verified.

For a given i, besides solution X_{iJ}, every of the other n solutions verify $x_{iJ} \geq 0$ to equality, and are proven to be affinely independent. Inequalities $x_{iJ} \geq 0$ are then facet-defining for P_{SCPKP}.

Any other inequality $x_{ij} \geq 0$, $j \neq J$ cannot be facet-defining. Indeed, $x_{ij} \geq 0$ can be obtained by summing $x_{ij'} \leq x_{ij'-1}$ for all $j' \in \{j+1, \ldots, J\}$ and $0 \leq x_{iJ}$. ■

Theorem 5

Inequalities $x_{ij} \leq x_{ij-1}$ are facet-defining for P_{SCPKP}.

Proof : As there are I groups and J elements, solutions X_{ij} and $X_{\emptyset}$ yield a total of $I \times J + 1 = n + 1$ different solutions, which are feasible otherwise (fd) is not verified.

Besides X_{ij-1}, each of the n solutions verifies $x_{ij} \leq x_{ij-1}$ to equality, and is proven to be affinely independent. Inequalities $x_{ij} \leq x_{ij-1}$ are then facet-defining for P_{SCPKP}.

■

Contrary to trivial inequalities with bound at 0 and the chain precedence constraints which always contain facet-defining inequalities, the bounds at 1 require a minimum capacity C to contain facet-defining inequalities.

Theorem 6

Inequalities $x_{ij} \leq 1$ are facet-defining for P_{SCPKP} if and only if $j = 1$ and $C \geq \sum_{j=1}^{J} W_j + W_1$.

Proof : Consider a group i'. For each item (i, j), consider a solution X'_{ij}, similar to X_{ij}, with $x_{i'1} = 1$ if $i' \neq i$. As there are I groups and J elements, solutions X'_{ij} yields a total of $I \times J = n$ different solutions, which are feasible if $C \geq \sum_{j=1}^{J} W_j + W_1$ and (fd) are verified. Each of the n solutions verifies $x_{i'1} \leq 1$ to equality, and is proven to be affinely independent in the same manner as solutions X_{ij}. Inequalities $x_{i'1} \leq 1$ are then facet-defining for P_{SCPKP} if $C \geq \sum_{j=1}^{J} W_j + W_1$.

In the case where $C < \sum_{j=1}^{J} W_j + W_1$, clearly $x_{iJ} + x_{i'1} \leq 1$ is valid, for any $i \leq I$, $i' \leq I$ and $i \neq i'$ because of the symmetric weights. In which case inequality $x_{iJ} \leq 1$ is dominated and cannot be facet-defining in this case.

Hence, inequalities $x_{i1} \leq 1$ are facet-defining if and only if $C \geq \sum_{j=1}^{J} W_j + W_1$.

Any other inequality $x_{ij} \leq 1$, $j \neq 1$ cannot be facet-defining. Indeed, $x_{ij} \leq 1$ can be obtained by summing $x_{ij'} \leq x_{ij'-1}$ for all $j' \in \{2, \ldots, j\}$ and $x_{i1} \leq 1$. ■

Because of the symmetric weights, if a solution is feasible, then any symmetric solution with respect to the group indices is also feasible. Moreover, the symmetries also appear in the facet-defining inequalities of the SCPKP.

Property 5

If an inequality is facet-defining for the SCPKP, any of its symmetries is also facet-defining for the SCPKP.

Proof : If an inequality is facet-defining for the SCPKP, there are n affinely independent valid solutions verifying the inequality to equality. As the weights are symmetric with respect to the groups, if a solution is valid, then any permutation of groups yields another valid solution. Hence, one can prove any symmetry of a facet-defining inequality to also be facet-defining, as it suffices to deduce the n valid solutions following the same permutation of groups. These new n points are necessarily affinely independent as they all undergo the exact same permutation of groups. ■

Clearly, the result of this property is directly due to the polyhedral symmetry, defined in **Definition 4**. Like the BKP, the SCPKP features three types of facet-defining inequalities: the ones from the initial formulation, *binary inequalities* with 0-1 coefficients, and *integer inequalities*, with non-negative integer coefficients. In the following **Example 7** shows the convex hull of an instance of the SCPKP.

Example 7

Let (4, 3, [3, 4, 2], V , 9) be an instance of the SCPKP. The convex hull obtained with PORTA [24] contains the following inequalities of the formulation, proven to be facet-defining in ***Theorem 4*** *and* ***Theorem 5****.*

$$
\begin{aligned}
&x_{13} \geq 0, \quad x_{23} \geq 0, \quad x_{33} \geq 0, \quad x_{43} \geq 0,\\
&x_{11} \geq x_{12}, && x_{12} \geq x_{13},\\
&x_{21} \geq x_{22}, && x_{22} \geq x_{23},\\
&x_{31} \geq x_{32}, && x_{32} \geq x_{33},\\
&x_{41} \geq x_{42}, && x_{42} \geq x_{43}.
\end{aligned}
$$

As well as the following inequalities:

$$
\begin{aligned}
&x_{12} + x_{22} + x_{32} + x_{41} \leq 1,\\
&x_{12} + x_{22} + x_{31} + x_{42} \leq 1,\\
&x_{12} + x_{21} + x_{32} + x_{42} \leq 1,\\
&x_{11} + x_{22} + x_{32} + x_{42} \leq 1,\\
&x_{11} + 2x_{12} + x_{21} + 2x_{22} + x_{31} + 2x_{32} + x_{41} + 2x_{42} \leq 3.
\end{aligned}
$$

This instance does not verify the condition described in Theorem 6, as $9 < 3 + 4 + 2 + 3 = 12$*. Hence, none of the inequalities* $x_{i1} \leq 1$ *are facet-defining in this example.*

Note that for each facet-defining inequality, all its symmetries with respect to the group indices are also facet-defining.

In the article, the polyhedral study focuses on the binary inequalities through a structure to handle their symmetries.

4.4 Pattern inequalities

In this section we introduce new inequalities. We are interested in the faces defined by these inequalities, i.e., the set of points of the polytope P_{SCPKP} verifying these inequalities to equality. To handle the symmetries of the inequalities efficiently, we introduce a structure called pattern.

4.4.1 Definitions

Definition 9 (*Pattern*)

A pattern $\mathcal{P}$ is a collection of I sets $S_i(\mathcal{P}) \subseteq \{1,\dots,J\}$, $i \leq I$.

A set $S_i(\mathcal{P})$ contains the indices j of the items in a same group. The sets of a pattern are not ordered, meaning that a pattern represents any permutation of an item set of the SCPKP.

As the aim is to produce inequalities from the patterns, we define the variable sets associated with a pattern.

Definition 10 (*Variable set $\mathcal{X}$ associated with $\mathcal{P}$*)

A variable set $\mathcal{X} \subseteq \mathcal{V}$ is associated with pattern $\mathcal{P}$ and a permutation π of $\{1,\dots,I\}$ if:
$x_{ij} \in \mathcal{X} \Leftrightarrow j \in S_{\pi(i)}(\mathcal{P})$.

We denote $\chi(\mathcal{P})$ the set of all variable sets associated with $\mathcal{P}$. Note that $|\chi(\mathcal{P})|$ is in general exponential. For the remainder of **Section 4.4**, when referring to $\mathcal{X} \in \chi(\mathcal{P})$, we consider without loss of generality that π is the identity permutation π_{id} if not mentioned otherwise.

The cardinality of a pattern $\mathcal{P}$ is the cardinality of any variable set associated with $\mathcal{P}$.

Definition 11 (*$card(\mathcal{P})$*)

The cardinality of a pattern $\mathcal{P}$ is $card(\mathcal{P}) = \sum_{i \leq I} |S_i(\mathcal{P})|$.

The rank of a pattern $\mathcal{P}$ is the valid upper bound for the sum of variables in any variable set associated with $\mathcal{P}$.

Definition 12 (*$rank(\mathcal{P})$*)

The rank of a pattern $\mathcal{P}$ is

$$rank(\mathcal{P}) = \max_{\mathcal{X} \in \chi(\mathcal{P})} \left\{ \max \sum_{x_{ij} \in \mathcal{X}} x_{ij} : \textit{satisfying } (4.3.1) - (4.3.2) \right\}.$$

The rank of a pattern can be computed with a shortest path algorithm [11] as described is **Section 4.6**.

With $rank(\mathcal{P})$ and $\chi(\mathcal{P})$, we can define the inequalities of a pattern $\mathcal{P}$ as follows.

Definition 13 (*Pattern inequalities*)

The pattern inequalities associated with a pattern $\mathcal{P}$ are the following, for any $\mathcal{X} \in \chi(\mathcal{P})$:

$$\sum_{x_{ij} \in \mathcal{X}} x_{ij} \leq rank(\mathcal{P}). \qquad (pi(\mathcal{X}))$$

Property 6

Pattern inequalities are valid for P_{SCPKP}.

Proof : By definition of the rank, and because the weights are symmetric, all pattern inequalities are valid. ■

We show in the following property that pattern inequalities can capture binary inequalities that are captured neither by minimal cover inequalities nor by (lifted) MIC inequalities.

Property 7

Pattern inequalities dominate minimal cover inequalities as well as MIC inequalities and binary lifted MIC inequalities.

Proof : Clearly, any binary inequality with the tightest right hand side can be represented by a pattern inequality. Hence, any binary inequality is either captured by pattern inequalities, or dominated by a pattern inequality. Besides, the following counter-example (**Example 8**) exhibits a facet-defining inequality of the SCPKP which can be obtained neither by minimal cover inequalities nor by (lifted) MIC inequalities.

Example 8

Let (4, 4, [5, 4, 2, 2], V , 20) be an instance of the SCPKP. The following inequality is part of the convex hull obtained with PORTA [24]:

$$x_{11} + x_{12} + x_{21} + x_{24} + x_{32} + x_{33} + x_{42} + x_{44} \leq 3. \quad (i1)$$

We show that inequality $(i1)$ *does not correspond to any of the inequalities introduced in* ***Section 4.2*** *for the PKP.*

Inequality $(i1)$ *is not a minimal cover inequality [17]. The set of variables associated with* $(i1)$ *is* $\mathcal{X} = \{x_{11}, x_{12}, x_{21}, x_{24}, x_{32}x_{34}, x_{42}, x_{44}\}$. *Clearly, this set of variables is not a minimal cover, as both* x_{21} *and* x_{24} *are in this set, but* x_{22} *and* x_{23} *are not despite chain precedence constraints* $x_{22} \leq x_{21}$, $x_{23} \leq x_{22}$ *and* $x_{24} \leq x_{23}$.

Inequality $(i1)$ *is not a MIC inequality [80]. Indeed,* $\mathcal{X}$ *is not a MIC, as its subsets are MIC. For example,* $U_1 = \{x_{11}, x_{21}, x_{32}\}$ *is subset of* $\mathcal{X}$, *and is a MIC.*

Inequality $(i1)$ *cannot be obtained with the lifting procedure of [80]. In this lifting procedure the MIC inequality is lifted with variables having two successors in the MIC. However, by definition of a MIC, there cannot be two items in a MIC if there is a precedence constraint between them. In the case of the SCPKP, it means a MIC can only have 1 item per group. As precedence constraints are only between items in a same group, there cannot be an item with two successors in the MIC. Hence, this lifting does not apply, and cannot yield* $(i1)$.

Inequality $(i1)$ *cannot be obtained with the sequential lifting procedure of [64]. Any minimal cover* $U \subseteq \mathcal{X}$ *has cardinality 3, meaning* $|U| - 1 \leq 2$. *Indeed, the induced cover with minimum weight of cardinality 4 is* $U_2 = \{x_{11}, x_{21}, x_{32}, x_{42}\}$, *but is not minimal as* $U_1 \subset U_2$ *is also an induced cover. As coefficients* α_{ij} *are zero from* ***Property 1****, the right hand-side of a MIC inequality cannot change with this lifting. Hence, inequality* $(i1)$ *cannot be obtained with this sequential lifting procedure.*

One can define pattern $\mathcal{P} = \{\{1,2\},\{1,4\},\{2,3\},\{2,4\}\}$. *This pattern is of rank 3. With* $\mathcal{X} = \{x_{11}, x_{12}, x_{21}, x_{24}, x_{32}, x_{33}, x_{42}, x_{44}\} \in \mathcal{X}(\mathcal{P})$, *one can obtain inequality* $(i1)$. *Such a result is due to pattern inequalities being a very large family of inequalities. In* ***Section 3.2****, we define necessary facet-defining conditions for pattern inequalities.*

■

As for any set of variables there is a pattern, and vice-versa, these pattern inequalities cover all the binary inequalities of the SCPKP. Also, because $|\mathcal{X}(\mathcal{P})|$ can be exponential, each pattern is associated with

a number of pattern inequalities that can be exponential. As the number of patterns of an SCPKP is also exponential, we need to define the conditions for a pattern to lead to tight pattern inequalities.

Definition 14 (*Pattern-facet*)

A pattern $\mathcal{P}$ is a pattern-facet if for every $\mathcal{X} \in \chi(\mathcal{P})$, $(pi(\mathcal{X}))$ is facet-defining.

4.4.2 Necessary facet-defining conditions

In this section we define three necessary conditions for a pattern to be a pattern-facet. The first one is for a pattern to have at least one item in each of its sets.

Property 8 (*Condition (i): no empty group*)

Let $\mathcal{P}$ be a pattern. If $\mathcal{P}$ is a pattern-facet, then it verifies condition (i):

(i) *For every set $S_i(\mathcal{P}) \in \mathcal{P}$:* $|S_i(\mathcal{P})| \geq 1$

Proof : Let $\mathcal{P}$ be a pattern-facet of rank k and $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. Clearly, if $|\mathcal{X}| = k$, then the pattern inequality is dominated by the sum of the bound $x_{ij} \leq 1$ for all $x_{ij} \in \mathcal{X}$. Hence, in the following, we suppose $card(\mathcal{P}) = |\mathcal{X}| > k$.

Suppose $S_i(\mathcal{P})$ does not verify (i) for a given i, i.e., $S_i(\mathcal{P}) = \emptyset$. Let $\mathcal{X}' = \mathcal{X} \cup \{x_{iJ}\}$ be a variable set. Because x_{iJ} is the only variable of $\mathcal{X}'$ for group i, $s_{iJ}(\mathcal{X}') \geq s_{i'j'}(\mathcal{X}')$ for any $x_{i'j'} \in \mathcal{X}'$. Then, the following inequality is valid:

$$\sum_{x_{i'j'} \in \mathcal{X}} x_{i'j'} + x_{iJ} \leq k.$$

Indeed, when $x_{iJ} = 0$, this inequality is valid by the rank of $\mathcal{P}$. When $x_{iJ} = 1$, there cannot be k variables of $\mathcal{X}$ to 1 simultaneously. Otherwise, as $s_{iJ}(\mathcal{X}') \geq s_{i'j'}(\mathcal{X}')$ for any $x_{i'j'} \in \mathcal{X}'$, one could set x_{iJ} to 0, and any other variable of $\mathcal{X}'$ to 1. This might reduce the total weight. In the case it does, this would lead to another solution. Such a solution would have $k+1$ variables of $\mathcal{X}$ to 1, which contradicts the rank of $\mathcal{P}$. Therefore, this inequality is valid.

This inequality dominates $(pi(\mathcal{X}))$. Indeed, one could sum it with $-x_{iJ} \leq 0$ to obtain $(pi(\mathcal{X}))$.

Thus, a pattern $\mathcal{P}$ is a pattern-facet only if $\mathcal{P}$ verifies condition (i). ■

The idea of the following condition is that, for any $\mathcal{X} \in \chi(\mathcal{P})$, there is a feasible solution with $(pi(\mathcal{X}))$ to equality, and $x_{iJ} = 1$ for any group i.

Property 9 (*Condition (ii): selection of item J*)

Let $\mathcal{P}$ be a pattern of rank k, and $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. If $\mathcal{P}$ is a pattern-facet, then $\mathcal{P}$ verifies condition (ii) :

(ii) *For each $i \leq I$, there is $\mathcal{Y} \subseteq \mathcal{V}$ a k-intersection of $\mathcal{X}$ with $x_{iJ} \in \mathcal{Y}$*

Proof : Let $\mathcal{P}$ be a pattern of rank k, and $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. Suppose there is an $i \leq I$ such that (ii) is not verified for i. This means that there is no feasible solution with k variables of $\mathcal{X}$ to 1, with $x_{iJ} = 1$. Therefore, the following inequality is valid:

$$\sum_{x_{i'j'} \in \mathcal{X}} x_{i'j'} + x_{iJ} \leq k.$$

Indeed, when $x_{iJ} = 0$, the inequality is valid by the rank of $\mathcal{P}$. When $x_{iJ} = 1$, the inequality is valid as there cannot be more than $k-1$ variables of $\mathcal{X}$ to 1, which sums to a total of at most k.

This inequality dominates the inequality $(pi(\mathcal{X}))$. Indeed, one could sum it with $-x_{iJ} \leq 0$ to obtain $(pi(\mathcal{X}))$.

Thus, a pattern $\mathcal{P}$ is a pattern-facet only if $\mathcal{P}$ verifies condition (ii). ■

The following condition is quite similar to condition (ii), but for any variable x_{ij-1} with $x_{ij} \in \mathcal{X}$, instead of any variable x_{iJ}.

Property 10 (*Condition (iii): independence of an item from its predecessor*)

Let $\mathcal{P}$ be a pattern of rank k, and $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. If $\mathcal{P}$ is a pattern-facet, then $\mathcal{P}$ verifies condition (iii) :

(iii) *For each variable $x_{ij} \in \mathcal{X}$, there is $\mathcal{Y} \subseteq \mathcal{V}$ a k-intersection of $\mathcal{X}$ with $x_{ij-1} \in \mathcal{Y}$ and $x_{ij'} \notin \mathcal{Y}$ for every $j' \geq j$.*

Proof : Let $\mathcal{P}$ be a pattern of rank k. Let $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. Suppose for some (i,j), $x_{ij} \in \mathcal{X}$ does not verify condition (iii). This means that there is no feasible solutions with a total of k variables of $\mathcal{X}$ to 1, with $x_{ij-1} = 1$ and $x_{ij} = 0$. Therefore, the following inequality is valid:

$$\sum_{x_{i'j'} \in \mathcal{X}} x_{i'j'} + x_{ij-1} - x_{ij} \leq k.$$

Indeed, when $x_{ij} = x_{ij-1} = 1$ or $x_{ij} = x_{ij-1} = 0$, this inequality is valid by the rank of $\mathcal{P}$. When $x_{ij-1} = 1$ and $x_{ij} = 0$, the inequality is valid as there cannot be more than $k-1$ variables of $\mathcal{X}$ to 1, which sums to at most k.

This inequality dominates the inequality $(pi(\mathcal{X}))$. Indeed, one could sum it with $-x_{ij-1} + x_{ij} \leq 0$ (equivalent to $x_{ij} \leq x_{ij-1}$) to obtain $(pi(\mathcal{X}))$.

Thus, a pattern $\mathcal{P}$ is a pattern-facet only if $\mathcal{P}$ verifies condition (iii). ■

For a given pattern $\mathcal{P}$, conditions (i) can clearly be verified in linear time. Also, conditions (ii) and (iii) can be verified in polynomial time. More precisely, it requires to solve the shortest path algorithm described in **Section 4.6** at most once for each variable. As these conditions are necessary, we define a flexible pattern, which verifies all of these three conditions.

Definition 15 (*Flexible pattern*)

A pattern $\mathcal{P}$ is a flexible pattern if it verifies conditions (i), (ii) and (iii).

The patterns are said to be flexible, as for $\mathcal{X} \in \chi(\mathcal{P})$, any variable of $\mathcal{X}$ can be set to 1 when $(pi(\mathcal{X}))$ is to equality. Hence, there is no restriction on which variable can be set to 1 when $(pi(\mathcal{X}))$ is to equality. Clearly, all binary facet-defining inequalities must be flexible pattern inequalities, as conditions (i), (ii) and (iii) are necessary. In particular, pattern inequalities dominate UMIC inequalities as proven in **Property 7**.

The following example illustrates conditions (i), (ii) and (iii) on inequality $(i1)$ from **Example 8**.

Example 9

Let (4, 4, [5, 4, 2, 2], V , 20) be an instance of the SCPKP. The following inequality is part of the convex hull description obtained with PORTA [24]:

$$x_{11} + x_{12} + x_{21} + x_{24} + x_{32} + x_{34} + x_{42} + x_{44} \leq 3. \qquad (i1)$$

We show that inequality $(i1)$ is a flexible pattern inequality.

Let $\mathcal{X} = \{x_{11}, x_{12}, x_{21}, x_{24}, x_{32}, x_{34}, x_{42}, x_{44}\}$ be a variable set. Clearly, $\mathcal{X}$ verifies condition (i).

We now verify if condition (ii) holds. For $i = 1$, variable set $\{x_{11}, x_{12}, x_{14}, x_{21}\}$ is a 3-intersection of $\mathcal{X}$. For $i = 2$, variable set $\{x_{11}, x_{21}, x_{24}\}$ is a 3-intersection of $\mathcal{X}$. For $i = 3$, variable set $\{x_{11}, x_{32}, x_{34}\}$ is a 3-intersection of $\mathcal{X}$. For $i = 4$, variable set $\{x_{11}, x_{42}, x_{44}\}$ is a 3-intersection of $\mathcal{X}$. Hence, condition (ii) holds.

We now verify if condition (iii) holds. For x_{11}, variable set $\{x_{21}, x_{32}, x_{42}\}$ is a 3-intersection of $\mathcal{X}$. For x_{12}, variable set $\{x_{11}, x_{21}, x_{24}\}$ is a 3-intersection of $\mathcal{X}$. For x_{21}, variable set $\{x_{11}, x_{32}, x_{34}\}$ is a 3-intersection of $\mathcal{X}$. For x_{24}, variable set $\{x_{11}, x_{12}, x_{21}, x_{23}\}$ is a 3-intersection of $\mathcal{X}$. For x_{32}, variable set $\{x_{11}, x_{12}, x_{21}, x_{31}\}$ is a 3-intersection of $\mathcal{X}$. For x_{34}, variable set $\{x_{11}, x_{12}, x_{32}, x_{33}\}$ is a 3-intersection of $\mathcal{X}$. For x_{42} (resp. x_{44}) one can reuse the 3-intersection for x_{32} (resp. x_{34}) by inverting groups 3 and 4. Hence, condition (iii) holds.

Consequently, inequality $(i1)$ is a pattern inequality associated with a flexible pattern.

The conditions on a flexible pattern $\mathcal{P}$ are not sufficient for $\mathcal{P}$ to be a pattern-facet. However, a minimum dimension can be guaranteed for the faces defined by flexible pattern inequalities.

4.4.3 Lower bound on the dimension of the faces defined by flexible pattern inequalities

In this section, we consider a flexible pattern $\mathcal{P}$ and a variable set $\mathcal{X} \in \chi(\mathcal{P})$. The idea of the following property is that for any x_{ij} $j < J$ (resp. x_{iJ}) there is a valid solution with $x_{ij} = 1$, $x_{ij+1} = 0$ (resp. $x_{iJ} = 1$) and $(pi(\mathcal{X}))$ to equality. In a sense it is a generalization of condition (ii) defined only for variables x_{iJ}, and condition (iii) defined only for variables x_{ij} such that $x_{ij+1} \in \mathcal{X}$.

Property 11 (*Generalization of (ii) and (iii) for any item of the SCPKP*)

Let $\mathcal{P}$ be a flexible pattern and $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. For any item (i, j), there is $\mathcal{Y} \subseteq \mathcal{V}$ a k-intersection of $\mathcal{X}$ with $x_{ij} \in \mathcal{Y}$ and $x_{ij'} \notin \mathcal{Y}$ for every $j' > j$.

The complete proof is in **B.1.2**, as it merely extends the proofs for conditions (ii) and (iii).

Conditions (i), (ii) and (iii) are necessary for a pattern $\mathcal{P}$ to be a pattern-facet. Moreover, with $\mathcal{X} \in \chi(\mathcal{P})$, the following theorem provides a lower bound on the number of linearly independent points verifying inequalities $(pi(\mathcal{X}))$ to equality when these three conditions are verified.

Theorem 7 ($n - card(\mathcal{P})$ *linearly independent points*)

Let $\mathcal{P}$ be a flexible pattern. Let $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. Let n be the number of variables of the SCPKP. There are at least $n - card(\mathcal{P})$ linearly independent points that verify inequality $(pi(\mathcal{X}))$ to equality.

Proof : Let $\mathcal{P}$ be a pattern of rank k, and $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. **Property 11** stipulates that if $\mathcal{P}$ is a flexible pattern, then for any item (i, j) there is a k-intersection $\mathcal{Y}_{ij} \subseteq \mathcal{V}$ of $\mathcal{X}$; $x_{ij} \in \mathcal{Y}_{ij}$; $x_{ij'} \notin \mathcal{Y}_{ij}$ for each $j' > j$ and $X_{\mathcal{Y}_{ij}}$ is feasible. Consider $X_{\mathcal{Y}_{ij}}$ for each variable $x_{ij} \notin \mathcal{X}$. Because $card(\mathcal{P}) = |\mathcal{X}|$, there are $n - card(\mathcal{P})$ solutions. We can prove that $X_{\mathcal{Y}_{ij}}$ is the only solution with $x_{ij} = 1$ and $x_{ij+1} = 0$. Let $x_{ij} \notin \mathcal{X}$ and $x_{i'j'} \notin \mathcal{X}$ be two distinct variables. We assumed without loss of generality in **Property 11** that $\mathcal{Y}_{i'j'} \setminus \{x_{i'j'}\} \subseteq \mathcal{X}$. As $x_{ij} \notin \mathcal{X}$, there would be a contradiction if $X_{\mathcal{Y}_{i'j'}}$ had $x_{ij} = 1$ and $x_{ij+1} = 0$.

Solutions $X_{\mathcal{Y}_{ij}}$ for variables $x_{ij} \notin \mathcal{X}$ are linearly independent, and proven to be valid in **Property 11**. As $\mathcal{Y}_{ij}$ is a k-intersection of $\mathcal{X}$, all these solutions also verify $(pi(\mathcal{X}))$ to equality. Hence, for a flexible pattern $\mathcal{P}$, there are $n - card(\mathcal{P})$ linearly independent points verifying $(pi(\mathcal{X}))$ to equality with $\mathcal{X} \in \mathcal{P}$. ■

Theorem 7 provides a lower bound on the dimensions of the faces defined by flexible pattern inequalities. Recall that for a pattern to be a flexible pattern it solely requires to verify conditions (i), (ii) and (iii). It is shown in **Section 4.6** that verifying if these three conditions hold for a given pattern can be done in polynomial time, and **Theorem 7** is used in the experimental results in **Section 4.7**.

The following section provides properties complementary to **Theorem 7**.

4.4.4 Properties of the lower sub-patterns

In this section, we first introduce new families of patterns, namely the sub-patterns and the lower sub-patterns. We then derive multiple properties for the lower sub-patterns associated with a flexible pattern. These properties are used in **Section 4.4.5**, to prove that necessary conditions (i) (ii) and (iii) are also sufficient for a family of flexible patterns to be a pattern-facet.

Definition 16 (*Sub-pattern*)

A pattern $\mathcal{P}'$ is sub-pattern of pattern $\mathcal{P}$ if there is a permutation π such that $S_{\pi(i)}(\mathcal{P}') \subseteq S_i(\mathcal{P})$ for all $i \leq I$.

If $\mathcal{P}$ is sub-pattern of $\mathcal{P}'$, then $\mathcal{P}'$ is *super-pattern* of $\mathcal{P}$. For the remainder of **Section 4.4**, we consider without loss of generality that the required permutation π is the identity permutation π_{id}.

We present a new set of sub-patterns for $\mathcal{P}$. In the following, properties are presented to show that these new patterns have large sub-patterns in common, thus inducing similarities. These similarities will be convenient to provide linearly independent points in the polytope P_{SCPKP}.

Definition 17 (*Lower sub-patterns $\mathcal{Q}_i$*)

Let $\mathcal{P}$ be a pattern of rank k. For a given $i \leq I$, a lower sub-pattern $\mathcal{Q}_i$ of $\mathcal{P}$ is such that $S_i(\mathcal{Q}_i) = S_i(\mathcal{P})$, $card(\mathcal{Q}_i) = k$, minimizing the sum of the set weights $s(\mathcal{Y})$ with $\mathcal{Y} \in \mathcal{X}(\mathcal{Q}_i)$.

Example 10

Let (4, 3, [3,4,2], V, 9) be an instance of the SCPKP. Let $\mathcal{P} = \{\{1,3\},\{2\},\{2\},\{2\}\}$ be a pattern of rank 2. Lower sub-pattern $\mathcal{Q}_1$ is $\{\{1,3\},\emptyset,\emptyset,\emptyset\}$, as $S_1(\mathcal{P}) = 2 = rank(\mathcal{P})$. As $W_1 < W_1 + W_2$, lower sub-pattern $\mathcal{Q}_2$ is $\{\{1\},\{2\},\emptyset,\emptyset\}$, and cannot be $\{\emptyset,\{2\},\{2\},\emptyset\}$ by definition. Similarly, lower sub-pattern $\mathcal{Q}_3$ (resp. $\mathcal{Q}_4$) is $\{\{1\},\emptyset,\{2\},\emptyset\}$ (resp. $\{\{1\},\emptyset,\emptyset,\{2\}\}$).

Let $\mathcal{P}$ be a pattern and $\mathcal{X} \in \mathcal{X}(\mathcal{P})$ be a variable set. Let $\mathcal{Q}_i$ be a lower sub-pattern of $\mathcal{P}$ and $\mathcal{Y} \in \mathcal{X}(\mathcal{Q}_i)$ be a variable set. By construction, if $x_{ij} \in \mathcal{Y}$, then for each variable $x_{ij'} \in \mathcal{X}$ with $j' \leq j$, $x_{ij'} \in \mathcal{Y}$, justifying hence the name lower sub-patterns. Also, patterns $\mathcal{Q}_i$ can be obtained via a shortest path algorithm defined in **Section 4.6**.

The idea now is to define minimum size of the sets of all lower sub-patterns. This is used in **Section 4.4.5** to prove that they have a common sub-pattern of cardinality $k-1$. This in turn is convenient to provide affinely independent points. For the remainder of **Section 4.4.4**, we consider a flexible pattern $\mathcal{P}$.

Lemma 1

Let $\mathcal{P}$ be a flexible pattern of rank k, and $\mathcal{X} \in \mathcal{X}(\mathcal{P})$ be a variable set. Let $\mathcal{Q}_i$ be a lower sub-pattern of $\mathcal{P}$, and $\mathcal{Y} \in \mathcal{X}(\mathcal{Q}_i)$ be a variable set. Variable set $\mathcal{Y} \cup \{x_{iJ}\}$ is a k-intersection of $\mathcal{X}$.

Proof : As $\mathcal{P}$ verifies condition (ii), there is a variable set that is a k-intersection of $\mathcal{X}$ containing x_{iJ}. By definition $\mathcal{Q}_i$ minimizes the sum of its set weights and all variables of group i in $\mathcal{X}$ are in $\mathcal{Y}$. Hence, if condition (ii) for variable x_{iJ} cannot be verified with $\mathcal{Y}$, there is a contradiction as $\mathcal{Q}_i$ cannot be minimizing the sum of its set weights. ■

Remark 1

Let $\mathcal{Y} \in \mathcal{X}(\mathcal{Q}_i)$ be a variable set. As $\mathcal{Y} \cup \{x_{iJ}\}$ is a k-intersection of $\mathcal{X}$, then $\mathcal{Y}$ is also a k-intersection of $\mathcal{X}$. Indeed, if $x_{iJ} \in \mathcal{X}$ then $x_{iJ} \in \mathcal{Y}$ by definition of $\mathcal{Q}_i$.

For the next properties in **Section 4.4.4** and **4.4.5**, we consider for the sake of simplicity that the sets of $\mathcal{P}$ are ordered such that $|S_i(\mathcal{P})| \leq |S_{i+1}(\mathcal{P})|$. Therefore, $S_1(\mathcal{P})$ is a smallest set of $\mathcal{P}$, and $\mathcal{Q}_1$ is the lower sub-pattern of $\mathcal{P}$ associated with $S_1(\mathcal{P})$. We define $\overline{U} = |S_I(\mathcal{P})|$ and $\underline{U} = |S_1(\mathcal{P})|$. Also, we define $S_i(\mathcal{P})(u)$ the u^{th} lowest index of $S_i(\mathcal{P})$ and $S_i(\mathcal{P})[u]$ the u^{th} highest index of $S_i(\mathcal{P})$.

Example 11

Let $\mathcal{P} = \{S_1(\mathcal{P}) = \{4\}, S_2(\mathcal{P}) = \{2,5\}, S_3(\mathcal{P}) = \{1,2\}, S_4(\mathcal{P}) = \{1,3,4\}\}$ be a pattern. In this case, $\overline{U} = 3$ and $\underline{U} = 1$. Also, $S_4(\mathcal{P})(1) = 1$, $S_4(\mathcal{P})(2) = 3$ $S_4(\mathcal{P})(3) = 4$, $S_4(\mathcal{P})[1] = 4$, $S_4(\mathcal{P})[2] = 3$ and $S_4(\mathcal{P})[3] = 1$.

Recall that the aim is to provide a lower bound on the size of each set of the lower sub-patterns. We first start by providing a lower bound on the size of each set of sub-pattern $\mathcal{Q}_1$ of $\mathcal{P}$.

Property 12 (*Minimum size on the sets of $\mathcal{Q}_1$*)

Let $\mathcal{P}$ be a flexible pattern of rank k. Lower sub-pattern $\mathcal{Q}_1$ of $\mathcal{P}$ is such that for every $i \leq I$, $S_i(\mathcal{Q}_1)$ contains the $|S_i(\mathcal{P})| - \underline{U}$ smallest indices of $S_i(\mathcal{P})$.

Remark 2

For any $i \leq I$ with $|S_i(\mathcal{P})| > \underline{U}$, it is equivalent to say that $S_i(\mathcal{Q}_1)$ contains the $|S_i(\mathcal{P})| - \underline{U}$ smallest indices of $S_i(\mathcal{P})$ and $S_i(\mathcal{P})[\underline{U}+1] \in S_i(\mathcal{Q}_1)$. In the following, the latter notation will be used.

The idea of the proof is illustrated by **Example 12** and **Figure 4.3**. In **Figure 4.3**, column i starting from the left, and row j, starting from the bottom represent variable x_{ij}, in the same manner as in **Figure 1.6**. When cell (i,j) is highlighted in (dark) gray, it means variable x_{ij} is contained in the corresponding variable set. The proof is in **B.1.3**

Example 12

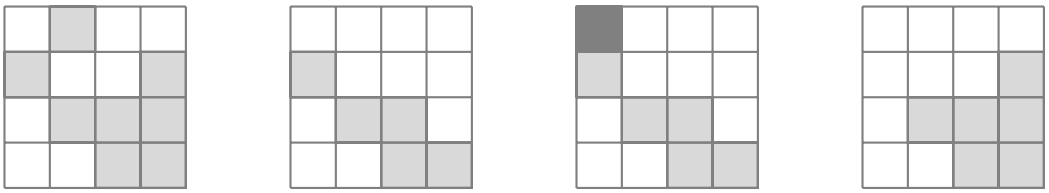
*Let $(4,5,w,V,C)$ be an instance of the SCPKP. Let $\mathcal{P} = \{S_1(\mathcal{P}) = \{3\}$, $S_2(\mathcal{P}) = \{2,4\}$, $S_3(\mathcal{P}) = \{1,2\}$, $S_4(\mathcal{P}) = \{1,2,3\}\}$ be a flexible pattern of rank 5. Let $\mathcal{Q}_1 = \{S_1(\mathcal{Q}_1) = \{3\}$, $S_2(\mathcal{Q}_1) = \{2\}$, $S_3(\mathcal{Q}_1) = \{1,2\}$, $S_4(\mathcal{Q}_1) = \{1\}\}$ be a lower sub-pattern of $\mathcal{P}$. In this case, $S_4(\mathcal{P})[\underline{U}+1] = 2 \notin S_4(\mathcal{Q}_1)$, hence $\mathcal{Q}_1$ does not verify **Property 12**. We show in the following that it leads to a contradiction with the rank of $\mathcal{P}$.*

*Let $\mathcal{X} = \{x_{13}, x_{22}, x_{24}, x_{31}, x_{32}, x_{41}, x_{42}, x_{43}\} \in \mathcal{X}(\mathcal{P})$ and $\mathcal{Y} = \{x_{13}, x_{22}, x_{31}, x_{32}, x_{41}\} \in \mathcal{X}(\mathcal{Q}_1)$ be variable sets as illustrated respectively in **Figure 4.3a** and **4.3b**. Let $\mathcal{Y}' = \mathcal{Y} \cup \{x_{14}\}$ be a variable set as illustrated in **Figure 4.3c**. From **Lemma 1** solution $X_{\mathcal{Y}'}$ is feasible. One can create a set $\mathcal{Y}'' = \mathcal{Y}' \setminus \{x_{13}, x_{14}\} \cup \{x_{42}, x_{43}\}$ as illustrated in **Figure 4.3d**. By removing $\{x_{13}, x_{14}\}$, there are no remaining variables in group 1, thus reducing the weight by $W_1 + W_2 + W_3 + W_4$. And as $x_{41} \in \mathcal{Y}'$, adding $\{x_{42}, x_{43}\}$ only increases the weight by $W_2 + W_3$. As the weights are non-negative, clearly $W_2 + W_3 \leq W_1 + W_2 + W_3 + W_4$, hence $X_{\mathcal{Y}''}$ is feasible. However, $\mathcal{Y}''$ is a $k+1$-intersection of $\mathcal{X}$. Indeed, since $\mathcal{Q}_1$ does not verify **Property 12**, only $\underline{U} = 1$ variable of $\mathcal{Y}' \setminus \mathcal{Y}''$ is in $\mathcal{X}$, namely x_{13} and $\underline{U}+1 = 2$ variables of $\mathcal{Y}'' \setminus \mathcal{Y}'$ are in $\mathcal{X}$, namely x_{42} and x_{43}. As $X_{\mathcal{Y}''}$ is valid, there is a contradiction with the rank of $\mathcal{P}$.*

Property 13 provides dependencies between the indices of $S_1(\mathcal{P})$ and any set $S_i(\mathcal{P})$, based on **Property 12**.

Property 13 (*Minimum indices for the smallest set*)

Let $\mathcal{P}$ be a flexible pattern and $i \in \{2,\dots,I\}$ be an index. For any $u \in \{1,\dots,\underline{U}\}$, if $|S_i(\mathcal{P})| \geq u + \underline{U}$ then $S_1(\mathcal{P})[u] \geq S_i(\mathcal{P})[u + \underline{U}]$.



(a) Set $\mathcal{X}$ (b) Set $\mathcal{Y}$ (c) Set $\mathcal{Y}'$ (d) Set $\mathcal{Y}''$

Figure 4.3: Illustration of **Example 12**

Proof : The proof is divided in two possibles cases, each being supported by a lemma. In the case $S_i(\mathcal{Q}_1) \subseteq S_i(\mathcal{P})$ with $S_i(\mathcal{P})[u] \in S_i(\mathcal{Q}_1)$, the proof is provided by **Lemma 5**. In the case $S_i(\mathcal{Q}_1) \subset S_i(\mathcal{P})$ with $S_i(\mathcal{P})[u] \notin S_i(\mathcal{Q}_1)$, the proof is provided by **Lemma 6**. Hence, the property is always verified. The two lemmas are in **B.1.4**. ■

For now, we provided a lower bound on the size of the sets of $\mathcal{Q}_1$. We also need similar bounds for the sets of any $\mathcal{Q}_i$, $i \leq I$, in order to characterize pattern facets. This result is given in **Property 14**, based on **Property 13**. Note that, it is not a generalization of **Property 12**. Indeed, **Property 12** addresses only $\mathcal{Q}_1$ and not any $\mathcal{Q}_i$, but with a larger minimal bound than the one in **Property 14**.

Property 14 (*Minimum size of the sets of any $\mathcal{Q}_i$*)

Let $\mathcal{P}$ be a flexible pattern. For every $i \leq I$, the lower sub-pattern $\mathcal{Q}_i$ of $\mathcal{P}$ is such that for a given $i' \leq I$ if $|S_{i'}(\mathcal{P})| \geq 2\underline{U}$ then $S_{i'}(\mathcal{P})[2\underline{U}] \in S_{i'}(\mathcal{Q}_i)$.

Proof : The proof is divided in three possible cases, each being supported by a lemma. In the case $|S_1(\mathcal{Q}_1)| + |S_i(\mathcal{Q}_1)| = |S_1(\mathcal{Q}_i)| + |S_i(\mathcal{Q}_i)|$, the proof is provided by **Lemma 7**. In the case $|S_1(\mathcal{Q}_i)| = 0$, the proof is provided by **Lemma 8**. In the case $|S_1(\mathcal{Q}_i)| > 0$, the proof is provided by **Lemma 9**. Hence, the property is always verified. The three lemmas are in **B.1.5** ■

The previous conditions are valid for any pattern. However, in the special case where pattern $\mathcal{P}$ is with $|S_1(\mathcal{P})| = 1$, **Property 14** indicates that $S_{i'}(\mathcal{P})[2] \in S_{i'}(\mathcal{Q}_i)$ for any $i' \leq I$ and $i \leq I$. Thus, the shape of all patterns $\mathcal{Q}_i$ is very restricted, as each set $S_{i'}(\mathcal{Q}_i)$ has at most one missing index set in comparison to $S_{i'}(\mathcal{P})$.

Example 13

*Let $\mathcal{P} = \{S_1(\mathcal{P}) = \{3\}, S_2(\mathcal{P}) = \{1,3,4\}, S_3(\mathcal{P}) = \{1,2,4\}, S_4(\mathcal{P}) = \{1,2,3,4\}\}$ be a flexible pattern. Pattern $\mathcal{P}$ verifies **Property 13**, as $S_1(\mathcal{P})[1] = 3$ is greater or equal to $S_2(\mathcal{P})[2] = 3$, $S_3(\mathcal{P})[2] = 2$ and $S_4(\mathcal{P})[2] = 3$. Consider $rank(\mathcal{P}) = 9$.*

*Consider the lower sub-pattern $\mathcal{Q}_1 = \{S_1(\mathcal{Q}_1) = \{3\}, S_2(\mathcal{Q}_1) = \{1,3,4\}, S_3(\mathcal{Q}_1) = \{1,2\}, S_4(\mathcal{Q}_1) = \{1,2,3\}\}$. Pattern $\mathcal{Q}_1$ verifies **Property 12**, as $\underline{U} = 1$ and there is at most $\underline{U} = 1$ missing index per set compared to $\mathcal{P}$.*

*Consider the lower sub-pattern $\mathcal{Q}_2 = \{S_1(\mathcal{Q}_2) = \emptyset, S_2(\mathcal{Q}_2) = \{1,3,4\}, S_3(\mathcal{Q}_2) = \{1,2,3\}, S_4(\mathcal{Q}_2) = \{1,2,3\}\}$. Pattern $\mathcal{Q}_2$ verifies **Property 14**, as $\underline{U} = 1$ and there is at most $2\underline{U} - 1 = 1$ missing index per*

set compared to $\mathcal{P}$.

This restricted shape on the lower sub-pattern is used in the following section to prove necessary and sufficient conditions for patterns containing a set of cardinality 1.

4.4.5 Necessary and sufficient conditions for patterns with a set of cardinality 1

In this section, we focus on patterns with at least one set of cardinality 1, hence we define for this section $\mathcal{P}$ a flexible pattern of rank k and with $\underline{U} = 1$. It is proven with **Property 14** that for such a pattern $\mathcal{P}$, its lower sub-patterns $\mathcal{Q}_i$ have a restricted shape. Using this result, we will show that the lower sub-patterns share many elements in common.

As mentioned in **Lemma 7**, for a given i, multiple lower sub-patterns $\mathcal{Q}_i$ with the exact same set of weights can exist. From now on we only consider for each $i \leq I$ the unique $\mathcal{Q}_i$ verifying the following tie-break rule.

Definition 18 (*Tie-break rule*)

Consider a flexible pattern $\mathcal{P}$ and a lower sub-pattern $\mathcal{Q}_i$ of $\mathcal{P}$. For any two indices $i' < i'' \leq I$ different from i, if

$$\sum_{j=S_{i'}(\mathcal{P})[2]+1}^{S_{i'}(\mathcal{P})[1]} W_j = \sum_{j=S_{i''}(\mathcal{P})[2]+1}^{S_{i''}(\mathcal{P})[1]} W_j;$$

then $S_{i''}(\mathcal{P})[1] \in S_{i''}(\mathcal{Q}_i)$ only if $S_{i'}(\mathcal{P})[1] \in S_{i'}(\mathcal{Q}_i)$.

From **Property 14**, at most one index is missing in a set of $\mathcal{Q}_i$ compared to $\mathcal{P}$. Hence, with such a rule there can only be one $\mathcal{Q}_i$ for a given i.

Example 14

Let $(4, 4, [2, 1, 1, 1], V, C)$ be an instance of the SCPKP. Let $\mathcal{P} = \{S_1(\mathcal{P}) = \{3\}, S_2(\mathcal{P}) = \{1, 3\}, S_3(\mathcal{P}) = \{2, 4\}, S_4(\mathcal{P}) = \{2, 4\}\}$ be a flexible pattern of rank 5. In this case, $W_2 + W_3 = W_3 + W_4 = 2$, meaning that there are 3 possible lower sub-pattern $\mathcal{Q}_1$ with the exact same weight: $\{\{3\}, \{1, 3\}, \{2\}, \{2\}\}$; $\{\{3\}, \{1\}, \{2, 4\}, \{2\}\}$; $\{\{3\}, \{1\}, \{2\}, \{2, 4\}\}$. The rule stipulates that $4 \in S_4(\mathcal{Q}_1)$ only if $4 \in S_3(\mathcal{Q}_1)$ and $3 \in S_2(\mathcal{Q}_1)$. Also, the rule stipulates that $4 \in S_3(\mathcal{Q}_1)$ only if $3 \in S_2(\mathcal{Q}_1)$. Only the first option for $\mathcal{Q}_1$ verifies the rule, and consequently is the only one considered.

To prove that the lower sub-patterns $\mathcal{Q}_i$ share many elements, we provide a pattern $\mathcal{C}$, sub-pattern to all $\mathcal{Q}_i$. We then prove that $\mathcal{C}$ is of cardinality $k - 1$.

Definition 19 (*Common sub-pattern $\mathcal{C}$ of all $\mathcal{Q}_i$*)

Let $\mathcal{P}$ be a flexible pattern, and for any $i \leq I$, let $\mathcal{Q}_i$ be the unique lower sub-pattern for i considering the tie-break rule. The pattern $\mathcal{C}$ is the largest cardinality pattern such that $\mathcal{C}$ is sub-pattern to all $\mathcal{Q}_i$ with $i \leq I$.

Property 15 (*Cardinality of $\mathcal{C}$*)

Let $\mathcal{P}$ be a flexible pattern, and $\mathcal{Q}_i$ be the unique lower sub-pattern considering the tie-breaking rule, for all $i \leq I$. The cardinality of $\mathcal{C}$ is $card(\mathcal{C}) = k - 1$.

An example of **Property 15** is provided in **Example 15**.

Proof : As $\mathcal{P}$ is a flexible pattern, **Property 14** holds, hence for any i and i' in $\{1,\dots,I\}$ such that $|S_{i'}(\mathcal{P})| \geq 2$, $S_{i'}(\mathcal{P})[2] \in S_{i'}(\mathcal{Q}_i)$. In other words, there is at most one element of $S_{i'}(\mathcal{P})$ not in $S_{i'}(\mathcal{Q}_i)$. Let k' be an integer equal to $k - (card(\mathcal{P}) - I)$. Let $\mathcal{L}$ be the set of the k' indices i' such that $S_{i'}(\mathcal{Q}_i) = S_{i'}(\mathcal{P})$. By definition of $\mathcal{Q}_i$, $i \in \mathcal{L}$. Because $S_{i'}(\mathcal{P})[2] \in S_{i'}(\mathcal{Q}_i)$ for any $i' \in \{1,\dots,I\}$, the $k' - 1$ sets $i' \neq i$ such that $S_{i'}(\mathcal{Q}_i) = S_{i'}(\mathcal{P})$ are the ones minimizing:

$$\sum_{j=S_{i'}(\mathcal{P})[2]+1}^{S_{i'}(\mathcal{P})[1]} W_j.$$

Consequently, $\mathcal{L}$ contains the $k' - 1$ indices minimizing this sum. Indeed, either i is in these $k' - 1$ indices, hence $\mathcal{L}$ contains the k' indices minimizing this sum, or i is not in these $k' - 1$ indices, but by construction $\mathcal{L}$ contains these $k' - 1$ indices.

As such, all patterns $\mathcal{Q}_i$ are all super-pattern of common pattern $\mathcal{C}$, with $card(\mathcal{C}) = (card(\mathcal{P}) - I) + k' - 1 = k - 1$. ■

Example 15

*Let $(3, 4, [2, 1, 3, 2], V, C)$ be an instance of the SCPKP. Let $\mathcal{P} = \{S_1(\mathcal{P}) = \{3\}, S_2(\mathcal{P}) = \{1, 2\}, S_3(\mathcal{P}) = \{1, 3\}, S_4(\mathcal{P}) = \{1, 3, 4\}\}$ be a flexible pattern of rank 6. First we identify the lower sub-pattern $\mathcal{Q}_1$. By definition, $S_1(\mathcal{Q}_1) = S_1(\mathcal{P}) = \{3\}$. From **Property 14**, as $\underline{U} = 1$, for any $i \leq 4$, $S_i(\mathcal{P})[2] \in S_i(\mathcal{Q}_1)$. In this case, $1 \in S_2(\mathcal{Q}_1)$, $1 \in S_3(\mathcal{Q}_1)$ and $1, 3 \in S_4(\mathcal{Q}_1)$. By definition, $card(\mathcal{Q}_1) = 6$, but only five elements have been identified yet. As $W_2 = 1 < W_4 = 2 < W_2 + W_3 = 4$ and $\mathcal{Q}_1$ minimize the sum of the set weights of $\mathcal{Y} \in \mathcal{X}(\mathcal{Q}_1)$, then clearly $2 \in S_2(\mathcal{Q}_1)$. In this case:*

$\mathcal{Q}_1 = \{S_1(\mathcal{Q}_1) = \{3\}, S_2(\mathcal{Q}_1) = \{1, 2\}, S_3(\mathcal{Q}_1) = \{1\}, S_4(\mathcal{Q}_1) = \{1, 3\}\}$

With a similar process, we also deduce:

$\mathcal{Q}_2 = \{S_1(\mathcal{Q}_2) = \emptyset, S_2(\mathcal{Q}_2) = \{1, 2\}, S_3(\mathcal{Q}_2) = \{1\}, S_4(\mathcal{Q}_2) = \{1, 3, 4\}\}$

$\mathcal{Q}_3 = \{S_1(\mathcal{Q}_3) = \emptyset, S_2(\mathcal{Q}_3) = \{1, 2\}, S_3(\mathcal{Q}_3) = \{1, 3\}, S_4(\mathcal{Q}_3) = \{1, 3\}\}$

$\mathcal{Q}_4 = \{S_1(\mathcal{Q}_4) = \emptyset, S_2(\mathcal{Q}_4) = \{1, 2\}, S_3(\mathcal{Q}_4) = \{1\}, S_4(\mathcal{Q}_4) = \{1, 3, 4\}\}$

There is $\mathcal{C} = \{\{S_1(\mathcal{C}) = \emptyset, S_2(\mathcal{C}) = \{1, 2\}, S_3(\mathcal{C}) = \{1\}, S_4(\mathcal{C}) = \{1, 3\}\}$ of cardinality 5 that is sub-pattern to all the aforementioned lower sub-patterns $\mathcal{Q}_i$. Note that only $S_2(\mathcal{C}) = S_2(\mathcal{P})$, which is because $s_{22}(\mathcal{X}) < s_{44}(\mathcal{X}) < s_{33}(\mathcal{X}) < s_{13}(\mathcal{X})$ and by definition the lower sub-patterns $\mathcal{Q}_i$ minimize the set weights of $\mathcal{Y} \in \mathcal{X}(\mathcal{Q}_i)$.

For the following results, we need to generalize the definition of lower sub-pattern.

Definition 20 (*Generalized lower sub-patterns $\mathcal{Q}_i(u)$*)

Let $\mathcal{P}$ be a pattern of rank k. For a given $i \leq I$ and $u \in \{0,\dots,|S_i(\mathcal{P})|\}$, the generalized lower sub-pattern

$\mathcal{Q}_i(u)$ of $\mathcal{P}$ is such that $card(\mathcal{Q}_i(u)) = k$ and $S_i(\mathcal{Q}_i(u))$ contains exactly the u smallest indices of $S_i(\mathcal{P})$, minimizing the sum of set weights $s(\mathcal{Y})$ with $\mathcal{Y} \in \mathcal{X}(\mathcal{Q}_i(u))$.

As for lower sub-patterns $\mathcal{Q}_i$, we can also find similarities between lower sub-patterns $\mathcal{Q}_i(u)$.

Property 16 (*Common elements between $\mathcal{C}$ and $\mathcal{Q}_i(u)$*)

Let $\mathcal{P}$ be a flexible pattern of rank k. For each $i \leq I$ and $u \in \{0,\dots,|S_i(\mathcal{P})|-1\}$, lower sub-pattern $\mathcal{Q}_i(u)$ of $\mathcal{P}$ is such that for any $i' \neq i$, $S_{i'}(\mathcal{C}) \subseteq S_{i'}(\mathcal{Q}_i(u))$.

Proof : By definition $card(\mathcal{Q}_i(u)) = k$ and $|S_i(\mathcal{Q}_i(u))| = u$. From **Property 15** $card(\mathcal{C}) = k-1$, and from **Property 14** for any $i' \leq I$ with $|S_{i'}(\mathcal{P})| \geq 2$, $S_{i'}(\mathcal{P})[2] \in S_{i'}(\mathcal{C})$. Let K be the difference between $card(\mathcal{Q}_i(u) \setminus S_i(\mathcal{Q}_i)(u))$ and $card(\mathcal{C} \setminus S_i(\mathcal{C}))$, i.e.,

$$\begin{aligned} K &= card(\mathcal{Q}_i(u) \setminus S_i(\mathcal{Q}_i)(u)) - card(\mathcal{C} \setminus S_i(\mathcal{C})), \\ &= (k-u) - (k-1-|S_i(\mathcal{C})|) = 1 + |S_i(\mathcal{C})| - u. \end{aligned}$$

Note that $u \leq |S_i(\mathcal{C})|$, meaning that $K \geq 1$. Because for each $i' \leq I$, $S_{i'}(\mathcal{P})[2] \in S_{i'}(\mathcal{C})$, and both $\mathcal{C}$ and $\mathcal{Q}_i(u)$ are sub-patterns of $\mathcal{P}$, then there are K sets such that $|S_{i'}(\mathcal{Q}_i(u))| = |S_{i'}(\mathcal{C})| + 1 = |S_{i'}(\mathcal{P})|$.

Let i' be the index of one of these K sets. By definition, $card(\mathcal{Q}_{i'}) = k$ and $card(\mathcal{C}) = k-1$, meaning that $card(\mathcal{Q}_{i'} \setminus S_{i'}(\mathcal{Q}_{i'})) = card(\mathcal{C} \setminus S_{i'}(\mathcal{C}))$. By definition $\mathcal{C}$ is sub-pattern of $\mathcal{Q}_{i'}$, hence $\mathcal{Q}_{i'} \setminus S_{i'}(\mathcal{Q}_{i'}) = \mathcal{C} \setminus S_{i'}(\mathcal{C})$. Consequently, $S_i(\mathcal{Q}_{i'}) = S_i(\mathcal{C})$ and we deduce:

$$\begin{aligned} & card(\mathcal{Q}_i(u) \setminus S_i(\mathcal{Q}_i)(u)) - card(\mathcal{Q}_{i'} \setminus S_i(\mathcal{Q}_{i'})), \\ =& (k-u) - \Big(card(\mathcal{Q}_{i'}) - |S_i(\mathcal{Q}_{i'})|\Big), \\ =& (k-u) - \Big(k - |S_i(\mathcal{C})|\Big) = K-1. \end{aligned}$$

From **Property 14** and because $\underline{U} = 1$, there are $K-1$ sets of $\mathcal{Q}_i(u)$ with one more element than the respective set of $\mathcal{Q}_{i'}$. However:

$$\begin{aligned} K-1 &= 1 + |S_i(\mathcal{C})| - u - 1, \\ &= |S_i(\mathcal{C})| - u, \\ &= |S_i(\mathcal{Q}_{i'})| - |S_i(\mathcal{Q}_i(u))|. \end{aligned}$$

The difference between $|S_i(\mathcal{Q}_{i'})|$ and $|S_i(\mathcal{Q}_i(u))|$ is exactly $K-1$. From **Property 14** if $i' \leq I$ is such that $|S_{i'}(\mathcal{P})| \geq 2$, $S_{i'}(\mathcal{P})[2] \in S_{i'}(\mathcal{C})$ and $\mathcal{C}$ is a sub-pattern of $\mathcal{Q}_{i'}$. Hence, there are $K-1$ indices i'' such that $|S_{i''}(\mathcal{Q}_i(u))| = |S_{i''}(\mathcal{Q}_{i'})| + 1$.

Let $\mathcal{L}$ be a set of indices, containing i and the $K-1$ aforementioned indices i''. Patterns $\mathcal{Q}_i(u)$ and $\mathcal{Q}_{i'}$ are such that

$$\sum_{i'' \in \mathcal{L}} |S_{i''}(\mathcal{Q}_i(u))| = \sum_{i'' \in \mathcal{L}} |S_{i''}(\mathcal{Q}_{i'})|.$$

By definition, both patterns $\mathcal{Q}_i(u)$ and $\mathcal{Q}_{i'}$ are of cardinality k, hence

$$\sum_{i'' \notin \mathcal{L}} |S_{i''}(\mathcal{Q}_i(u))| = \sum_{i'' \notin \mathcal{L}} |S_{i''}(\mathcal{Q}_{i'})|.$$

As both lower sub-patterns minimize their set weights we deduce that for any $i'' \notin \mathcal{L}$, $S_{i''}(\mathcal{Q}_i(u)) = S_{i''}(\mathcal{Q}_{i'})$.

Hence, for all $i'' \neq i$, $S_{i''}(\mathcal{Q}_{i'}) \subseteq S_{i''}(\mathcal{Q}_i(u))$. As $\mathcal{C}$ is a sub-pattern of $\mathcal{Q}_i$, then for each $i'' \neq i$, $S_{i''}(\mathcal{C}) \subseteq S_{i''}(\mathcal{Q}_i(u))$. ■

With all lower sub-patterns $\mathcal{Q}_i(u)$ defined, the following theorem shows that a flexible pattern with a set of cardinality 1 is a pattern-facet.

Theorem 8

Let $\mathcal{P}$ be a pattern with $\underline{U} = 1$. $\mathcal{P}$ is a pattern-facet if and only if $\mathcal{P}$ is a flexible pattern.

Proof : Recall that without loss of generality, pattern sets can be ordered such that $|S_i(\mathcal{P})| \leq |S_{i+1}(\mathcal{P})|$, meaning $|S_1(\mathcal{P})| = 1$. Let $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. Consider the following lower sub-patterns: $\mathcal{Q}_1$, $\mathcal{Q}_i$ for every $i \leq I$ such that $S_i(\mathcal{P})[1] \notin S_i(\mathcal{Q}_1)$; $\mathcal{Q}_{i'}(u)$ for every $i' \leq I$ and $u \in \{0,..,|S_{i'}(\mathcal{C})|-1\}$. For all mentioned sub-patterns, we consider their respective variable set denoted $\mathcal{X}_1$; $\mathcal{X}_i$; $\mathcal{X}_{i'}(u)$ and their respective solution $X_{\mathcal{X}_1}$; $X_{\mathcal{X}_i}$; $X_{\mathcal{X}_{i'}(u)}$. This results in a total of $card(\mathcal{P})$ solutions. There is one solution $X_{\mathcal{X}_1}$. As $\mathcal{P}$ is of rank k, there are $card(\mathcal{P}) - k$ solutions $X_{\mathcal{X}_i}$. As $card(\mathcal{C}) = k - 1$, there are $k - 1$ solutions $X_{\mathcal{X}_{i'}(u)}$.

By definition of lower sub-patterns $\mathcal{Q}_i$ and $\mathcal{Q}_i(u)$ and from **Property 11**, all mentioned variable sets are k-intersections of $\mathcal{X}$. Hence, all mentioned solutions are feasible and verify $(pi(\mathcal{X}))$ to equality.

Consider now the points associated with the afore-enumerated solutions. We can prove these points to be linearly independent. Start by considering first the point associated with $X_{\mathcal{X}_1}$. As it is the only point considered, it is necessarily linearly independent. From **Property 15**, $\mathcal{C}$ is sub-pattern to all $\mathcal{Q}_i$ and of cardinality $k - 1$. Consequently, $\mathcal{Q}_i$ with $S_i(\mathcal{P})[1] \notin S_i(\mathcal{Q}_1)$ is the only lower sub-pattern, excluding the generalized lower sub-patterns, with $S_i(\mathcal{P})[1] \in S_i(\mathcal{Q}_i)$. It results that for each solution $X_{\mathcal{X}_i}$, the associated point is the only one with $x_{iS_i(\mathcal{P})[1]} = 1$, thus they are linearly independent. For each solution $X_{\mathcal{X}_{i'}(u)}$, starting with large u, the associated point is the first one with $x_{i'S_{i'}(\mathcal{P})[u]} = 0$, thus being linearly independent.

The enumerated solutions yield $card(\mathcal{P})$ linearly independent points. Moreover, from **Theorem 7**, for each variable $x_{ij} \notin \mathcal{X}$ there is a feasible solution with $x_{ij} = 1$ and $x_{ij+1} = 0$ that verifies $(pi(\mathcal{X}))$ to equality. Sequentially adding these points associated with their corresponding solutions to our pool of $card(\mathcal{P})$ points still keeps them linearly independent, as there are the only ones with $x_{ij} = 1$ and $x_{ij+1} = 0$ with $x_{ij} \notin \mathcal{X}$. As there are $n - card(\mathcal{P})$ of these new points, there is a total of n linearly independent points.

Thus, a pattern $\mathcal{P}$ with a set of cardinality 1 is a pattern-facet if and only if it is a flexible pattern. ■

Recall that for a pattern to be a flexible pattern it solely requires to verify conditions (i), (ii) and (iii). Hence, a pattern with a set of cardinality 1 is a pattern-facet if conditions (i), (ii) and (iii) hold. It is shown in **Section 4.6** that verifying if these three conditions hold for a given pattern can be done in polynomial time. As for **Theorem 7**, the result of **Theorem 8** is used for the experimental results in **Section 4.7**. Indeed, this theorem guarantees a flexible pattern $\mathcal{P}$ to be a pattern-facet if $\min_{i \leq I} |S_i(\mathcal{P})| = 1$, or provides a lower bound on the dimensions of the faces defined by the pattern inequalities of $\mathcal{P}$ otherwise.

4.4.6 Conditions for any pattern

For a pattern $\mathcal{P}$ such that $\min_{i \leq I} |S_i(\mathcal{P})| \geq 2$, the conditions (i), (ii) and (iii) are necessary but not sufficient for $\mathcal{P}$ to be a pattern-facet. This is because the lower sub-patterns $\mathcal{Q}_i$ of $\mathcal{P}$ lose many of their structural properties in the general case. In the following, we present three new conditions, that will complement the aforementioned conditions. For this purpose, we present new patterns $\mathcal{R}_u$ that will take the role of the lower sub-patterns $\mathcal{Q}_i$. The idea is to help constructing points with a common coefficient, which is convenient to provide independent points.

In the following, we consider $\mathcal{P}$ a flexible pattern of rank k.

Definition 21 (*Nested sub-patterns $\mathcal{R}_u$*)

Consider a flexible pattern $\mathcal{P}$ of rank k. Patterns $\{R_u, 1 \leq u \leq U'\}$ are nested sub-patterns of $\mathcal{P}$ if for each u, $card(\mathcal{R}_u) = k - u$, $\mathcal{R}_u$ is a lower sub-pattern of $\mathcal{P}$ and $\mathcal{R}_u$ sub-pattern of $\mathcal{R}_{u-1}$.

For the following, we define $U' = \min_{u \leq k}\{\max_{i \leq I}\{S_i(\mathcal{P}) - S_i(\mathcal{R}_u)\} \geq u\}$. U' is illustrated in **Example 16**. We consider the following subset of nested sub-patterns $\{\mathcal{R}_u, 1 \leq u \leq U'\}$. We also consider $\mathcal{X} \in \chi(\mathcal{P})$ and $\mathcal{Y}_u \in \chi(\mathcal{R}_u)$, for each $u \leq U'$.

In the following we define three new conditions. Condition (iv) indicates that there are k-intersections of $\mathcal{X}$ with $\mathcal{Y}_u$ and u variables in a same group. Condition (v) is similar, but with u variables in at least two different groups. Condition (vi) is a more constrained version of condition (iii), but only for variables in $\mathcal{Y}_{U'}$.

Definition 22 (*Condition (iv): selection of items in the same group*)

Consider a flexible pattern $\mathcal{P}$ of rank k, with $\mathcal{X} \in \chi(\mathcal{P})$. For any $u \leq U'$ and $i \leq I$ such that $|S_i(\mathcal{P})| - |S_i(\mathcal{R}_u)| \geq u$, there is a variable set $\mathcal{Z}$ containing the u variables with the smallest indices of group i in $\mathcal{X} \setminus \mathcal{Y}_u$. The variable set $\mathcal{Z} \cup \mathcal{Y}_u$ is a k-intersection of $\mathcal{X}$.

Definition 23 (*Condition (v): selection of items in different groups*)

Consider a flexible pattern $\mathcal{P}$ of rank k, with $\mathcal{X} \in \chi(\mathcal{P})$. For any $u \in \{2, \ldots, U'\}$, there is $\mathcal{Z}_u \subset \mathcal{X}$ of cardinality u such that $\mathcal{Z}_u \cap \mathcal{Y}_1 = \emptyset$, $\mathcal{Z}_u$ contains variables in at least two different groups and the variable set $\mathcal{Y}_u \cup \mathcal{Z}_u$ is a k-intersection of $\mathcal{X}$.

Definition 24 (*Condition (vi): constrained independence of an item from its predecessor*)

For any $x_{ij} \in \mathcal{Y}_{U'}$, there is $\mathcal{Z} \subseteq \mathcal{V}$ a k-intersection of $\mathcal{X}$ such that for every variable $x_{i'j'} \in \mathcal{Y}_{U'}$, if $i' = i$ and $j' \geq j$, then $x_{i'j'} \notin \mathcal{Z}$, otherwise $x_{i'j'} \in \mathcal{Z}$.

Note that contrary to conditions (i), (ii) and (iii), conditions (iv), (v) and (vi) do not apply on $\mathcal{P}$ but, instead, on the nested sub-patterns. In the following example, we illustrate conditions (iv), (v) and (vi), and show that the nested sub-patterns used for these conditions are not necessarily the sub-patterns with the smallest weight.

Example 16

Let $(4,4,[8,4,2,3],V,37)$ *be an instance of the SCPKP. Let* $\mathcal{P} = \{S_1(\mathcal{P}) = \{1,2\}, S_2(\mathcal{P}) = \{2,3\}, S_3(\mathcal{P}) = \{2,4\}, S_4(\mathcal{P}) = \{2,4\}\}$ *be a flexible pattern of rank 4. Consider the sub-pattern* $\mathcal{R}_1 = \{S_1(\mathcal{R}_1) = \{1\}, S_2(\mathcal{R}_1) = \{2,3\}, S_3(\mathcal{R}_1) = \emptyset, S_3(\mathcal{R}_1) = \emptyset\}$. *In this case,* $|S_3(\mathcal{P})| - |S_3(\mathcal{R}_1)| = 2 > 1$. *Consequently,* U' *is greater than 1. Consider the sub-pattern* $\mathcal{R}_2 = \{S_1(\mathcal{R}_2) = \{1\}, S_2(\mathcal{R}_2) = \{2\}, S_3(\mathcal{R}_2) = \emptyset, S_3(\mathcal{R}_2) = \emptyset\}$. *In this case,* $|S_i(\mathcal{P})| - |S_i(\mathcal{R}_2)|$ *is at most 2 for each* $i \leq 4$, *hence* $U' = 2$. *Note that* $\mathcal{R}_2$ *is sub-pattern of* $\mathcal{R}_1$, *meaning that they are nested sub-patterns of* $\mathcal{P}$.

Let $\mathcal{X} = \{x_{11}, x_{12}, x_{22}, x_{23}, x_{32}, x_{34}, x_{42}, x_{44}\}$, $\mathcal{Y}_1 = \{x_{11}, x_{22}, x_{23}\}$ *and* $\mathcal{Y}_2 = \{x_{11}, x_{22}\}$ *be variable sets.*

Condition (iv) *is verified. For* $\mathcal{Y}_1$ *the following variable sets are 4-intersections of* $\mathcal{X}$: $\{x_{11}, x_{12}, x_{22}, x_{23}\}$, $\{x_{11}, x_{22}, x_{23}, x_{32}\}$, $\{x_{11}, x_{22}, x_{23}, x_{42}\}$. *For* $\mathcal{Y}_2$ *the following variable sets are 4-intersections of* $\mathcal{X}$: $\{x_{11}, x_{22}, x_{32}, x_{34}\}$, $\{x_{11}, x_{22}, x_{42}, x_{44}\}$.

Condition (v) *is verified. For* $\mathcal{Y}_2$ *the following variable set is a 4-intersection of* $\mathcal{X}$: $\{x_{11}, x_{12}, x_{22}, x_{32}\}$, *with* x_{12} *and* x_{32} *being in different groups, while not being in* $\mathcal{Y}_1$.

Condition (vi) *is verified. For* $\mathcal{Y}_2$, *the following variable sets are 4-intersections of* $\mathcal{X}$: $\{x_{11}, x_{12}, x_{32}, x_{34}\}$ *and* $\{x_{22}, x_{23}, x_{32}, x_{34}\}$. *The first one contains all variables of* $\mathcal{Y}_2$ *but* x_{22}, *and the second one contains all variables of* $\mathcal{Y}_2$ *but* x_{11}.

In this example, the lower sub-pattern $\mathcal{R}_1$ *is the unique one of cardinality 3 minimizing the set weights of* $\mathcal{Y}_1$. *However,* $\mathcal{R}_2$ *is not the one of cardinality 2 minimizing the set weights of* $\mathcal{Y}_2$. *In fact,* $\mathcal{R}'_2 = \{S_1(\mathcal{R}'_2) = \{1,2\}, S_2(\mathcal{R}'_2) = \emptyset, S_3(\mathcal{R}'_2) = \emptyset, S_3(\mathcal{R}'_2) = \emptyset\}$ *is of cardinality 2 and minimizes the set weights of* $\mathcal{Y}'_2 \in \mathcal{X}(\mathcal{R}'_2)$. *Because* $\mathcal{R}'_2$ *is not sub-pattern to* $\mathcal{R}_1$, *they cannot be nested sub-patterns.*

For a set of nested sub-patterns $\mathcal{R}_u$, conditions (iv) can be verified in polynomial time. Indeed, the k-intersection are explicitly defined, it only requires to compute the sum of its set weights. Also, verifying conditions (v) and (vi) requires to find a k-intersection, as for conditions (ii) and (iii). Hence, it can be verified in polynomial time, in a similar fashion as for conditions (ii) and (iii), using a variant of the shortest path algorithm. However, conditions (iv), (v) and (vi) apply on a set of nested sub-patterns, instead of on a single pattern $\mathcal{P}$. As shown in **Example 16**, the lower sub-patterns of $\mathcal{P}$ minimizing the sum of the set weights of their respective variable sets may not be nested. This makes conditions (iv), (v) and (vi) hard to verify simultaneously.

Remark 3

The complexity of finding nested sub-patterns $\mathcal{R}_u$, $u \in \{1,\dots,U'\}$ *of* $\mathcal{P}$, *verifying* (iv), (v) *and* (vi) *remains an open question.*

To the best of our knowledge, it seems to require an exponential enumeration. Indeed, there is no a priori information on which set of sub-patterns $\mathcal{R}_u$ of $\mathcal{P}$ will verify conditions (iv), (v) and (vi) and be nested simultaneously. Consequently, the best algorithm we found so far would be to enumerate all sub-patterns $\mathcal{R}_u$, $u \in \{1,\dots,U'\}$. This means enumerating all u-intersections, $u \in \{1,\dots,U'\}$ of $\mathcal{X} \in \mathcal{X}(\mathcal{P})$. In **Section 4.6**, we show that finding for any k, a k-intersection is equivalent to finding the shortest path of length k in a Directed Acyclic Graph (DAG). Hence, such algorithm would enumerate all shortest

paths of length at most U' in a DAG, which is exponential. For each $u \in \{1,\dots,U'\}$, there is generally an exponential number of sub-patterns R_u. In order to verify (iv), (v), and (vi), one needs to find patterns R_u, $u \in \{1,\dots,U'\}$, such that the sub-patterns R_u are nested. This requires to compare each sub-pattern R_u with all sub-patterns R'_u, $u' \neq u$, hence, comparing exponential number of sub-patterns.

The following theorem shows that conditions (i) to (vi) are sufficient pattern-facet conditions.

Theorem 9 (*Sufficient pattern-facet conditions*)

Let $\mathcal{P}$ be a flexible pattern. If nested sub-patterns $\{\mathcal{R}_u, 1 \leq u \leq U'\}$ of $\mathcal{P}$ verify conditions (iv), (v) and (vi), then $\mathcal{P}$ is a pattern-facet.

The complete proof is in **B.1.6**, relying on **Lemma 10** defined in the same appendix.

Even though conditions (iv), (v) and (vi) can provide pattern-facets, we will only be using flexible patterns for the experimental results. The rationale behind is that as stated after **Remark 3**, one may need to enumerate for an exponential computational time to verify (iv), (v) and (iv). Another reason is that from **Theorems 7** and **8**, a flexible pattern $\mathcal{P}$ is a pattern-facet if it has a set of cardinality one. Otherwise, it has a lower bound on the dimensions of the faces defined by the pattern inequalities of $\mathcal{P}$. As shown in **Section 4.7**, using flexible pattern, with or without the cuts of CPLEX, within a Branch and Cut (B&C) framework can drastically reduce the number of nodes explored and the computational time required to solve instances of the SCPKP. Furthermore, experimental results in **Section 4.7** show that for some instances, a few flexible patterns are generated. Consequently, adding conditions (iv), (v) and (vi) would further reduce the number of patterns generated, making it more difficult to measure their impact.

4.5 Separation of pattern inequalities

In order to use pattern inequalities in a B&C framework, we must define the separation problem.

Definition 25 (*Separation problem for pattern inequalities*)

Let (I, J, W, V, C) be an instance of the SCPKP. Consider $\widetilde{x}$ a fractional solution for this instance. The separation problem for pattern inequalities is to find the pattern $\mathcal{P}$ and the permutation π maximizing $\sum_{i=1}^{I} \sum_{j \in S_{\pi(i)}(\mathcal{P})} \widetilde{x}_{ij} - rank(\mathcal{P})$

We can prove that the separation problem for pattern inequalities is NP-hard for a family of patterns.

Definition 26 (*Diagonal pattern*)

A pattern $\mathcal{P}$ is diagonal if $|S_i(\mathcal{P})| \leq 1$ for every $i \leq I$ and if $j \in S_i(\mathcal{P})$ then $j \notin S_{i'}(\mathcal{P})$ for every $i \leq I$ and $i' \leq I$ such that $i \neq i'$.

Theorem 10

Separation problem for diagonal pattern inequalities is NP-hard.

Proof : We perform a reduction from the separation problem of extended cover inequalities [7] for the knapsack polytope, shown NP-hard in 13.

Consider a knapsack instance with capacity C', J' items with weights W'_i, $i \in \{1,\dots,J'\}$, and an associated fractional point $\widetilde{x}^{KP} \in [0,1]^{J'}$. The problem considered is to separate, with respect to point $\widetilde{x}^{KP}$, the extended-cover inequalities, which have the following form : for a given subset $K \subseteq \{1,...,J'\}$,

$$\sum_{j \in K} x_i \leq \rho(K)$$

where $\rho(K)$ is the knapsack rank of K, defined as the maximum number of items of K that can be packed into the knapsack without violating capacity C'.

Now consider the following instance of the separation problem for pattern inequalities of the SCPKP . Consider SCPKP instance (I, J, W, V, C). We consider $W_i = W'_i - W'_{i-1}$, $C = C'$, $I = J = J'$, and values V can be arbitrarily defined. Note that any diagonal pattern $\mathcal{P}$ of this SCPKP corresponds to a subset of knapsack items $K \subseteq \{1,...,J'\}$. By definition, the rank of this pattern is equal to $\rho(K)$, the knapsack rank of K. Now consider the following fractional point:

$$\widetilde{x}_{ij} = \begin{cases} \widetilde{x}^{KP}_i & \text{if } j \leq i \\ 0 & \text{otherwise} \end{cases}$$

Finding the most violated diagonal pattern inequality corresponds to finding a diagonal pattern $\mathcal{P}$ maximizing $\sum_{i=1}^{I} \sum_{j \in S_i(\mathcal{P})} \widetilde{x}_{ij} - rank(\mathcal{P})$. Let $\mathcal{P}$ be such a pattern. The associated set K of knapsack items is a solution to the extended-cover inequalities separation. Conversely, any solution K to the latter problem corresponds to a solution to this pattern inequalities separation problem. ■

This result gives the intuition that the separation problem for pattern inequalities is also NP-hard. We propose a conjecture on how this result could be proven.

Conjecture 1

Consider the knapsack problem instance and the SCPKP instance described in the proof of ***Theorem 10****. Consider the following fractional point: $\widetilde{x}_{ij} = \widetilde{x}^{KP}_i$ for any $j \leq J$. Finding the most violated pattern inequality corresponds to finding the most violated cover inequalities, with integer coefficients below or equal to J.*

We can however consider a special case of the separation problem for a fixed pattern. We then prove that this special case can be solved in polynomial time.

Definition 27 (*Fixed-pattern separation problem for pattern inequalities*)

Let (I, J, W, V, C) be an instance of the SCPKP. Consider $\widetilde{x}$ a fractional solution for this instance. Let $\mathcal{P}$ be a pattern of rank k. The fixed-pattern separation problem is to find the permutation π maximizing $\sum_{i=1}^{I} \sum_{j \in S_i(\mathcal{P})} \widetilde{x}_{\pi(i)j} - k$

As the pattern $\mathcal{P}$ is fixed for this special case, k is a constant and can be ignored. Hence, the aim of the fixed-pattern separation problem is to find, for a given pattern, the permutation π maximizing

$\sum_{i=1}^{I}\sum_{j\in S_i(\mathcal{P})}\widetilde{x}_{\pi(i)j}$. Even if the number of permutations π for a pattern is exponential (see **Section 3**), solving the fixed-pattern separation problem for a pattern $\mathcal{P}$ can be done by solving a Maximum Matching Problem (MMP).

Definition 28 (*Maximum Matching Problem*)

Let $\mathcal{H}$ be a weighted bipartite graph. The MMP is to find the set of edges E, such that at most one edge of E is incident to each vertex of $\mathcal{H}$, while maximizing the sum of the weight of E.

Property 17

Finding a permutation π maximizing the $\sum_{i=1}^{I}\sum_{j\in S_i(\mathcal{P})}\widetilde{x}_{\pi(i)j}$ can be obtained by solving an instance of the MMP.

Proof : The aim is to find the permutation π of the sets of $\mathcal{P}$, maximizing $\sum_{i=1}^{I}\sum_{j\in S_i(\mathcal{P})}\widetilde{x}_{\pi(i)j}$. To do so, we can build a bipartite graph $\mathcal{H}=(\mathcal{H}_1,\mathcal{H}_2,E)$. Each vertex in $\mathcal{H}_1$ corresponds to a set $S_{i'}(\mathcal{P})$ $i\in\{1,\dots,I\}$ and each vertex in $\mathcal{H}_2$ corresponds to a group, $i'\in\{1,\dots,I\}$. In the set of edges E there is an edge (i,i') for each $i\in\mathcal{H}_1$, $i'\in\mathcal{H}_2$. The edge (i,i') has a weight equal to $\sum_{j\in S_i(\mathcal{P})}\widetilde{x}'_{i'j}$

Solving this MMP to optimality yields a matching maximizing the weight of the considered edges. One can deduce a permutation from this matching: $\pi(i)=i'$ if edge (i,i') is in the matching. As the MMP is solved to optimality, permutation π is such that $\sum_{i=1}^{I}\sum_{j\in S_i(\mathcal{P})}\widetilde{x}_{\pi(i)j}$ is maximized. ■

The following example illustrates how solving an MMP provides the permutation maximizing the left-hand side of a pattern inequality.

Example 17

*Let $(4,4,W,V,C)$ be an instance of the SCPKP. Let $\widetilde{x}=[[1,1,0,0],[1,0.7,0.7,0],[0.6,0,0,0],[1,0,0,0]]$ be a fractional solution. Let $\mathcal{P}=\{S_1(\mathcal{P})=\{1,3\},\ S_2(\mathcal{P})=\{1,2\},\ S_3(\mathcal{P})=\{1\},\ S_4(\mathcal{P})=\{1\}\}$ be a pattern with $rank(\mathcal{P})=5$. **Table 4.1** represents the weight matrix of $\mathcal{H}$, with $\widetilde{X}[i]\in\mathcal{H}_1$ the vertex corresponding to the group i of $\widetilde{X}$.*

In this example, the optimal solution to the MMP is: $(S_1(\mathcal{P}),2)$, $(S_2(\mathcal{P}),1)$, $(S_3(\mathcal{P}),3)$, $(S_4(\mathcal{P}),4)$. This solution has a value $2+1.7+0.6+1=5.3$. The inequality with $\mathcal{X}\in\chi(\mathcal{P})$, ordered with respect to the permutation corresponding to the solution of the MMP is:

$$x_{11}+x_{12}+x_{21}+x_{23}+x_{31}+x_{41}\leq 5.$$

This inequality cuts the fractional point as the left-hand side of the inequality equals 5.3.

As the MMP is a problem that can be solved in polynomial time [61], so is the fixed-pattern separation problem.

	1	2	3	4
$S_1(\mathcal{P})$	1	1.7	0.6	1
$S_2(\mathcal{P})$	2	1.7	0.6	1
$S_3(\mathcal{P})$	1	1	0.6	1
$S_4(\mathcal{P})$	1	1	0.6	1

Table 4.1: Weight matrix

4.6 Two-phase Branch & Cut

In this section, we present algorithms in order to use the pattern inequalities within a B&C framework. The separation problem for pattern inequalities is proven to be NP-hard for a family of pattern in **Theorem 10**. Based on this theorem, we proposed **Conjecture 1** indicating that our intuition is that the separation problem is also NP-hard in general. and conditions (i) (ii) and (iii) can be verified independently from the B&C, we devise a two-phase B&C scheme. The first phase generates flexible patterns as a pre-processing. The second phase separates the associated pattern inequalities within a B&C framework.

The algorithm to generate flexible-patterns, described in **Section 4.6.2**, is based on two algorithms defined in **Section 4.6.1**. The separation algorithm, described in **Section 4.6.3**, is to produce the most violated inequality for a given pattern in polynomial time. The two-phase B&C scheme is described in **Section 4.6.4**

4.6.1 Graph model associated with variable sets

The shortest path problem and many of its variants are known to be easy to solve [11] [31] when there are no negative cost cycles. In the case of the SCPKP, for a given pattern $\mathcal{P}$, it is possible to define a graph associated with $\mathcal{X} \in \chi(\mathcal{P})$, for which solving a variant of the shortest path problem gives the rank of $\mathcal{P}$. This graph will be used to compute the rank of a pattern and to verify conditions (ii) and (iii). Such a graph could also be used to compute the sub-patterns $\mathcal{Q}_i$ or to verify conditions (v) and (vi).

Consider a pattern $\mathcal{P}$ and a variable set $\mathcal{X} \in \chi(\mathcal{P})$. Let $G_{SCPKP} = (V_{SCPKP}, A_{SCPKP})$ be the graph defined as follows. The variable set $\mathcal{X}$ is associated with the vertex set V_{SCPKP}, a source vertex p and a sink vertex q are added to V_{SCPKP}. For convenience purposes, each vertex in V_{SCPKP} is denoted by the corresponding variable in $\mathcal{X}$ and the vertices associated with $x_{ij} \in \mathcal{X}$ are renumbered with the same order in a compact sequence in $\mathcal{X}$. More precisely consider the following renumbering, denoted by parentheses on the indices: For any $x_{ij} \in \mathcal{X}$, we consider $x_{(i)(j)} = x_{ij}$ with $(i) = i$ and (j) the number of variables $x_{ij'} \in \mathcal{X}$ with $j' \leq j$. To each couple $(x_{(i)(j)}, x_{(i)(j+1)}) \in \mathcal{X}^2$ corresponds an arc in A_{SCPKP}. To each $x_{(i)(j)} \in \mathcal{X}$ corresponds an arc in A_{SCPKP} from $x_{(i)(j)}$ to $x_{(i')(1)}, i' > i$. To each $x_{(i)(1)} \in \mathcal{X}$ corresponds an arc from source vertex p to $x_{(i)(1)}$ in A_{SCPKP}. Similarly, for each vertex $x_{(i)(j)}$, it corresponds an arc from $x_{(i)(j)}$ to sink vertex q. Finally the weight of any arc heading to $x_{(i)(j)}$ is $s_{ij}(\mathcal{X})$ as defined in **Section 4.3.3** and to sink vertex q is 0. Note that graph G_{SCPKP} is a DAG with non-negative weights, meaning that shortest

path problems on G_{SCPKP} are easy to solve.

Example 18

*Let $(3,3,[3,2,2],V,C)$ be an instance of the SCPKP. For the variable set $\mathcal{X} = \{x_{11}, x_{12}, x_{21}, x_{22}, x_{23}, x_{31}, x_{33}\}$, the vertex set is $V_{SCPKP} = \{p, q, x_{(1)(1)}, x_{(1)(2)}, x_{(2)(1)}, x_{(2)(2)}, x_{(2)(3)}, x_{(3)(1)}, x_{(3)(2)}\}$. **Figure 4.4** illustrates graph G_{SCPKP} associated with $\mathcal{X}$.*

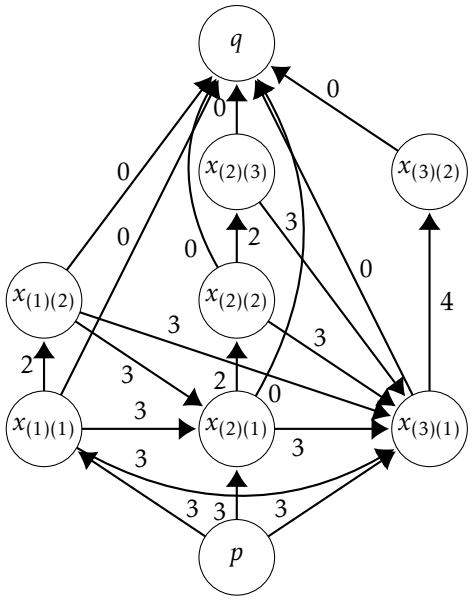


Figure 4.4: Graph G_{SCPKP} for **Example 18**

With graph G_{SCPKP} defined with respect to $\mathcal{X}$, the following property makes the link between a path in G_{SCPKP} and a variable set $\mathcal{Y} \subseteq \mathcal{X}$.

Property 18

Finding a shortest path in graph G_{SCPKP} featuring exactly $k+1$ arcs is equivalent to finding $\mathcal{Y} \subset \mathcal{X}$, with $\mathcal{Y}$ featuring the following properties: 1) $|\mathcal{Y}| = k$, 2) for every $x_{ij} \in \mathcal{Y}$ for each $x_{ij'} \in \mathcal{X}$ with $j' < j$, $x_{ij'} \in \mathcal{Y}$, 3) $\mathcal{Y}$ minimizes its set weights $s_{ij}(\mathcal{Y})$.

Proof : First, suppose that we found the shortest path in G_{SCPKP} with exactly $k+1$ arcs. 1) The path between p and q with exactly $k+1$ arcs is a path with $k+2$ vertices. Note that such a path necessarily exists with $k \in \{1,\dots,card(\mathcal{P})\}$. Indeed, by construction of G_{SCPKP}, there is a path going through all nodes $x_{(i)(j)}$ and there is an arc from any $x_{(i)(j)}$ to q. Also any arc $(x_{(i)(j)}, x_{(i')(j')})$ is with $(i') \geq i$ or $(j') > (j)$, thus there are no cycles in G_{SCPKP}. Hence, the $k+2$ vertices are necessarily different. As p and q do not correspond to any variables of $\mathcal{X}$, a path with exactly $k+1$ arcs represents a set $\mathcal{Y}$ with exactly k variables of $\mathcal{X}$

2) By construction of G_{SCPKP}, the only arc towards $x_{(i)(j)}$ is from $x_{(i)(j-1)}$ with $(j) > 1$. Consequently, for each $x_{ij} \in \mathcal{Y}$ such that $x_{ij'} \in \mathcal{X}$ and $j' < j$, then $x_{ij'} \in \mathcal{Y}$.

3) By construction of G_{SCPKP}, for each $x_{ij} \in \mathcal{X}$, every arc towards x_{ij} has a weight $s_{ij}(\mathcal{X})$. As such, a path with a set $\mathcal{Y} \subseteq \mathcal{X}$ of variables has a weight equal to $\sum_{x_{ij} \in \mathcal{Y}} s_{ij}(\mathcal{X})$. Hence, set $\mathcal{Y}$ corresponds to a shortest path minimizing the sum of its set weights.

Now, suppose that we have $\mathcal{Y} \subseteq \mathcal{X}$ satisfying 1), 2) and 3).

By construction of G_{SCPKP}, for each variable in $\mathcal{Y}$ there is a corresponding vertex in G_{SCPKP}. Besides, as $\mathcal{Y}$ satisfies 2), then there exists a path from p to q by sorting the variables $x_{ij} \in \mathcal{Y}$ by decreasing i and j. As $|\mathcal{Y}| = k$, the associated path has $k+1$ arcs. Finally, as $\mathcal{Y}$ verifies 3), then there are no other path in G_{SCPKP} with with exactly $k+1$ arcs with a lower value. ■

Computing the rank Clearly, if the sum of the set weights of $\mathcal{Y}$ is smaller than or equal to C, then $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$. By definition, if a k-intersection of $\mathcal{X}$ exists, then there is a feasible solution with k variables of $\mathcal{X}$ to 1. Hence, the idea to compute the rank of a pattern $\mathcal{P}$ is to find a maximum cardinality path in G_{SCPKP} associated with $\mathcal{X} \in \chi(\mathcal{P})$, with its total weight being smaller than or equal to C. From **Property 18**, if the maximum cardinality path found is with $k+1$ arcs, then the rank of $\mathcal{P}$ is k. Such a rank computing algorithm is described in **Algorithm 1**.

The **Algorithm 1** returns the rank of a pattern $\mathcal{P}$. In the following we detail **Algorithm 1**, which is illustrated in **Example 19**. In order to compute the rank, we use a variant of the Bellman-Ford algorithm [11] to compute the maximum cardinality path with a total weight less or equal C. The cardinality of such a path is then translated into the rank of $\mathcal{P}$. The first difference with the Bellman-Ford algorithm is that the vector $dist$ contains, at the end of the n^{th} iteration of the while loop, the distance between source p and any other node $v \in V_{SCPKP}$ in exactly n arcs. Hence, at each iteration of the while loop, $nDist$ is initialized with value ∞, and for all $v \in V_{SCPKP}$, $nDist[v]$ is replaced only if there is a path from p to v in n arcs. More precisely, the value $nDist[v]$ is replaced only if during the n^{th} iteration of the while loop there is a path from p to $v' \in V_{SCPKP}$ in $n-1$ arcs, i.e., $dist[n] < \infty$ and an arc from v' to v. The second difference is the stopping condition, which is when all paths have a distance greater than C. As all weights are positive, as soon as all paths have weight greater than C, there cannot be a path with even more arcs of total weight less or equal C. When the algorithm stops, there is no path with $nbArcs$ with total weight less or equal C. Hence, the maximum cardinality path found is with $nbArcs - 1$ arcs, containing $nbArcs - 2$ vertices different than p and q. Consequently, the rank of $\mathcal{P}$ is $k = nbArcs - 2$ as p and q do not represent any variable.

Example 19

*Let $(3, 3, [3, 2, 2], V, 6)$ be an instance of the SCPKP. Let $\mathcal{X} = \{x_{11}, x_{12}, x_{21}, x_{22}, x_{23}, x_{31}, x_{33}\}$ be the variable set as in **Example 18**, with its graph depicted in **Figure 4.4**. The table of values computed with **Algorithm 1** for this instance is depicted in **Table 4.2**, where the value of $nbArcs$ and $dist[v]$ for each $v \in V_{SCPKP}$ is shown for the initialization (first row), and at the end of each iteration of the while loop (second to fifth row). The algorithm stops when $nbArcs = 4$ as all values are greater than $C = 6$. Hence the maximum cardinality path is with $nbArcs - 1 = 3$ arcs, going through $nbArcs - 2$ vertices besides p and q. Consequently, the rank obtained in this example is 2.*

$nbArcs$	p	$x_{(1)(1)}$	$x_{(1)(2)}$	$x_{(2)(1)}$	$x_{(2)(2)}$	$x_{(2)(3)}$	$x_{(3)(1)}$	$x_{(3)(3)}$	q
0	0	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	3	∞	3	∞	∞	3	∞	∞
2	∞	∞	5	6	5	∞	6	7	3
3	∞	∞	∞	8	8	7	8	10	5
4	∞	∞	∞	∞	10	10	10	12	7

Table 4.2: Values of **Algorithm 1** relative to **Example 19**.

Algorithm 1 Computing the rank of a pattern

procedure computeRank(pattern $\mathcal{P}$):
  build $G_{SCPKP} = (V_{SCPKP}, A_{SCPKP})$ associated with $\mathcal{X} \in \chi(\mathcal{P})$ for permutation π_{id}
  $dist[p] \leftarrow 0$
  $dist[v] \leftarrow +\infty, \forall v \in V_{SCPKP} \setminus \{p\}$
  $nbArcs \leftarrow 0$
  while $\min(dist) \leq C$ **do**
    $nDist[v] \leftarrow \infty, \forall v \in V_{SCPKP}$
    for $a \in A_{SCPKP}$ **do**
      $nDist[a.head] \leftarrow \min(nDist[a.head], dist[a.tail] + a.weight)$
    end for
    $dist \leftarrow nDist$
    $nbArcs \leftarrow nbArcs + 1$
  end while
  $k \leftarrow nbArcs - 2$
  return k
end procedure

Verifying conditions (ii) **and** (iii) Let $\mathcal{P}$ be a pattern of rank k and $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set with permutation π_{id}. To verify if condition (ii) holds for a group i, one needs to find a set $\mathcal{Y}$ being a k-intersection of $\mathcal{X}$ with $x_{iJ} \in \mathcal{Y}$. To verify if condition (iii) holds for x_{ij}, one needs to find a set $\mathcal{Y}$ being a k-intersection of $\mathcal{X}$ with $x_{ij-1} \in \mathcal{Y}$ and $x_{ij} \notin \mathcal{Y}$. Let u be the number of variables $x_{ij'} \in \mathcal{X}$, with $j' \leq J$ for condition (ii), and with $j' < j$ for condition (iii). Finding such k-intersection is to find a shortest path of length $k - u + 1$ in G_{SCPKP}, without any variable of group i. The **Algorithm 2** returns a boolean, equals to True if (ii) is verified for x_{ij}, $j < J$, or (iii) is verified for x_{iJ}. To do so, we use another variant of the Bellman-Ford algorithm, to compute the path with minimal weight and cardinality $k - u + 1$. The proposed algorithm shares various similarities with **Algorithm 1** for managing vector $dist$ and $nDist$ in order to only keep, at the n^{th} iteration of the while loop, the shortest path between p and $v \in V_{SCPKP}$ with cardinality exactly n. The first difference with **Algorithm 1** lies in the initialization phase of **Algorithm 2**, where a second set $\mathcal{X}' = \mathcal{X} \setminus x_{ij'}$ for each $j' \in S_i(\mathcal{P})$. The second difference is the stopping condition. In **Algorithm 1**, we looked for the path with the largest cardinality, hence we stop when no path has a total weight less then or equal to C. In the case of **Algorithm 2** we do the opposite; we look for the path with minimum weight, for a given cardinality. Hence, we iterate until $dist$ contains the distances

of the minimum weight paths between p and all $v \in V_{SCPKP}$ with cardinality $k-u+1$, meaning with $k-u$ vertices different from p and q.

The **Algorithm 2**, for given integers j, i and a pattern $\mathcal{P}$ returns a boolean. In the case $j = J+1$, the algorithm returns true only if (ii) is verified for i, and in the case $j \in S_i(\mathcal{P})$, returns true only if (iii) is verified for i and j. Indeed, this algorithm searches for the path with minimum weight, and exactly $k-u+1$ arcs, between p and q. By construction of $\mathcal{X}'$, this path is without any variables of group i. Consequently if the total weight of the generated shortest path plus $\sum_{j'=1}^{j-1} W_{j'}$ is smaller than or equal to C, then there is a k-intersection of $\mathcal{X}$ as follows. The k-intersection contains all $k-u$ variables in the generated shortest path besides p and q and the variables $x_{ij'} \in \mathcal{V}$, $j' < j$. In the case $j \leq J$, the resulting k-intersection contains the variable x_{ij-1} but not x_{ij}, which corresponds to condition (iii). In the case $j = J+1$, the resulting k-intersection contains x_{iJ}, which corresponds to the condition (ii).

Remark 4

*Both algorithms described run in polynomial time with respect to $|V_{SCPKP}|$, with $|V_{SCPKP}| = card(\mathcal{P})+2$ in **Algorithm 1**, and $|V_{SCPKP}| = card(\mathcal{P}) + 2 - |S_i(\mathcal{P})|$ in **Algorithm 2**. In the case of **Algorithm 1**, the **while** loop can occur $|V_{SCPKP}|$ times, and in the case of **Algorithm 2**, the **while** loop can occur $rank(\mathcal{P}) \leq |V_{SCPKP}|$ times. In both cases, in the **while** loop we iterate over the arcs of A_{SCPKP}, meaning $|A_{SCPKP}| \leq |V_{SCPKP}|^2$ operations. Hence, both algorithms have a worst case time complexity of $\mathcal{O}(|V_{SCPKP}|^3)$. It is worth mentioning that because of the structure of G_{SCPKP}, $|A_{SCPKP}|$ is usually much smaller than $|V_{SCPKP}|^2$ as proven in **Property 19**. Hence, these algorithms usually require much less than $|V_{SCPKP}|^3$ operations.*

Property 19

Let $\mathcal{P}$ be a pattern and $\underline{U}$ be the cardinality of the smallest set $S_i(\mathcal{P})$. Let $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set and $G_{SCPKP} = (V_{SCPKP}, A_{SCPKP})$ be the graph associated with $\mathcal{X}$, then $\frac{|V_{SCPKP}|^2}{|A_{SCPKP}|} \approx \underline{U}$.

The proof is given in **B.1.7**.

Example 20

*Consider a pattern $\mathcal{P} = \{\{1,2\},\{1,2,3\},\{1,3\}\}$ with $\underline{U} = 2$. The following variable set is associated with $\mathcal{P}$, $\mathcal{X} = \{x_{11}, x_{12}, x_{21}, x_{22}, x_{23}, x_{31}, x_{33}\}$, as in **Example 18**. The vertex set is $V_{SCPKP} = \{p, q, x_{(1)(1)}, x_{(1)(2)}, x_{(2)(1)}, x_{(2)(2)}, x_{(2)(3)}, x_{(3)(1)}, x_{(3)(2)}\}$. **Figure 4.4** illustrates graph G_{SCPKP} associated with $\mathcal{X}$. One can compute that there are 7 arcs towards q, 3 arcs from p and 4 arcs $(x_{(i)(j)}, x_{(i)(j+1)})$. There are also 2 arcs $(x_{(1)(j)}, x_{(2)(1)})$ and $(x_{(1)(j)}, x_{(3)(1)})$ as well as 3 arcs $(x_{(2)(j)}, x_{(3)(1)})$. In total, $|A_{SCPKP}| = 21$, while $|V_{SCPKP}| = 9$. Clearly, $|A_{SCPKP}| < \frac{|V_{SCPKP}|^2}{\underline{U}} = \frac{81}{2} = 40.5$.*

Theorem 11

Verifying condition (i), (ii) and (iii) can be done in polynomial time.

Algorithm 2 Verifying conditions (ii) and (iii)

procedure VERIFYCONDITION(pattern $\mathcal{P}$, $i \in \{1,\dots,I\}$, $j \in \{1,\dots,J+1\}$):
▷ $j \leq J$ for condition (ii), $j = J+1$ for condition (iii)
  $u \leftarrow |\{j' \in S_i(\mathcal{P}) : j' < j\}|$
  $\mathcal{X} \leftarrow$ set in $\chi(\mathcal{P})$ for permutation π_{id}
  $\mathcal{X}' \leftarrow \mathcal{X} \setminus \{x_{ij'} : j' \in S_i(\mathcal{P})\}$
  build $G_{SCPKP} = (V_{SCPKP}, A_{SCPKP})$ associated with $\mathcal{X}'$
  $dist[p] \leftarrow 0$
  $dist[v] \leftarrow +\infty, \forall v \in V_{SCPKP} \setminus \{p\}$
  $nbArcs \leftarrow 0$
  while $nbArcs \leq rank(\mathcal{P}) - u$ **do**
    $nDist[v] \leftarrow \infty, \forall v \in V_{SCPKP}$
    for $a \in A_{SCPKP}$ **do**
      $nDist[a.head] \leftarrow \min(nDist[a.head], dist[a.tail] + a.weight)$
    end for
    $dist \leftarrow nDist$
    $nbArcs \leftarrow nbArcs + 1$
  end while
  return $dist[q] + \sum_{j'=1}^{j-1} W_{j'} \leq C$
end procedure

Proof : Condition (i) requires to verify if each set is non-empty, which is linear with respect to the number of sets I.

Condition (ii) requires to run **Algorithm 2** for each x_{iJ}, $i \leq I$, which is an algorithm of complexity $\mathcal{O}(|V_{SCPKP}|^3)$, $|V_{SCPKP}| \leq n$ a total of $I \leq n$ times. Condition (ii) can be verified in $\mathcal{O}(n^4)$ time.

Condition (iii) requires to run **Algorithm 2** for each $x_{ij} \in \mathcal{X}$, which is an algorithm of complexity $\mathcal{O}(|V_{SCPKP}|^3)$, $|V_{SCPKP}| \leq n$ a total of $|\mathcal{X}| \leq n$ times. Condition (iii) can be verified in $\mathcal{O}(n^4)$ time. ■

Remark 5

*It is worth mentioning that the time complexity $\mathcal{O}(n^4)$ is rarely reached. Indeed, only one pattern has a cardinality equal to n: the one containing I times the set $\{1,\dots,J\}$. All other patterns have a smaller cardinality, which reduces the required time to verify (ii) and (iii) in two ways. The first one is that **Algorithm 2** is needed $card(\mathcal{P})$ times for (iii), and $I \leq card(\mathcal{P})$ times for (ii). The second one is that the complexity of **Algorithm 2** depends on $|V_{SCPKP}|$, which is $card(\mathcal{P}) + 2 - |S_i(\mathcal{P})|$ as mentioned in **Remark 4**. Then likewise **Algorithm 2** usually requires much less than $|V_{SCPKP}|^3$ operations.*

Algorithm 2 can also be extended to gather further information on a pattern as described in the following. These extensions are not used for the experimental results in **Section 4.7**. Only **Algorithm 1** and **Algorithm 2** are required to apply **Theorem 7** and **Theorem 8**.

Computing generalized lower sub-patterns $\mathcal{Q}_i(u)$ To find a flexible pattern, there is no need to compute the lower sub-patterns $\mathcal{Q}_i(u)$. However, it is worth mentioning that finding $\mathcal{Q}_i(u)$ can also be done with a variant of the shortest path algorithm. More precisely, finding $\mathcal{Q}_i(u)$ can be obtained via **Algorithm 2**, for x_{ij+1} with $j = S_i(\mathcal{Q}_i(u))(u)$. The only modification is to add an extra step to memorize the paths, and to return the path to node q.

Verifying conditions (v) **and** (vi) Let $\mathcal{P}$ be a pattern and $\{\mathcal{R}_u, 1 \leq u \leq U'\}$ be the nested sub-patterns of $\mathcal{P}$. To verify if conditions (v) or (vi) hold for its nested sub-patterns $\mathcal{R}_u$, one needs to find k-intersections. Finding a k-intersection is also what is required to verify conditions (ii) and (iii). Hence, a similar algorithm as **Algorithm 2** can be used to verify (v) and (vi) for a given sub-pattern $\mathcal{R}_u$. However, as mentioned after **Remark 3**, finding a set of nested sub-patterns verifying (v) and (vi) may require to enumerate an exponential number of sub-patterns.

4.6.2 Pattern generation

The pattern generation procedure heavily relies on conditions (ii) and (iii) and also on the rank. The complete procedure is described in **Algorithm 3**, using **Algorithm 1** and **Algorithm 2**.

The aim of **Algorithm 3** is to generate a pattern verifying conditions (i) (ii) and (iii). To do so, we first generate a random integer k as a target rank for the pattern $\mathcal{P}$ to be constructed. Pattern $\mathcal{P}$ is initialized with I sets $\{J\}$. Then the goal is to iteratively modify pattern $\mathcal{P}$ until we obtain a flexible-pattern with rank k. To do so, we first aim to increase the cardinality of $\mathcal{P}$ until $card(\mathcal{P}) > k$. While the cardinality of $\mathcal{P}$ is less than k, a random group i such that $j > 1$ with j the smallest index in $S_i(\mathcal{P})$ is selected and we add $j-1$ to $S_i(\mathcal{P})$. Then, once $card(P) \geq k$, we update $\mathcal{P}$ in order to satisfy flexible-pattern conditions, in particular (ii) and (iii). To do so, while conditions (ii) and (iii) are not satisfied, we randomly select i and j such that $j \notin S_i(\mathcal{P})$ and $j+1 \in S_i(\mathcal{P})$ or $j = J$ (if such indices exist, otherwise the current iteration stops). Then we check conditions (ii) or (iii) for group i and index j with **Algorithm 2**. If it returns false, then in the case $j < J$ we replace $j+1$ by j in $S_i(\mathcal{P})$, and in the case $j = J$ we add J to $S_i(\mathcal{P})$.

In **Algorithm 3** the lower bound on k ensures that the rank is high enough to create a flexible pattern. Indeed, because of condition (i), too small of a rank will lead to patterns that cannot be flexible patterns. The reason why we initialize $\mathcal{P}$ with I groups of sets $\{J\}$ is for $\mathcal{P}$ to satisfy condition (i). Indeed, when modifying $\mathcal{P}$, either an element is added to a set, or an element is replaced. Hence the size of the sets never decreases, and there is at least 1 element in each group. In order for the pattern inequalities of $\mathcal{P}$ not to be trivial, it requires $card(\mathcal{P}) > rank(\mathcal{P})$. As pattern $\mathcal{P}$ is initialized with cardinality I, in the case $rank(\mathcal{P}) \geq I$ one needs to add elements to $\mathcal{P}$. For the purpose of this algorithm, we start with elements of higher indices. Note that **Algorithm 3** always ends. In the case where conditions (ii) and (iii) are satisfied by $\mathcal{P}$, then we exit loop at line 10. Otherwise, if none of the elements in $\mathcal{P}$ can be considered to verify if condition (ii) or (iii) holds, then we exit the loop at line 14.

Algorithm 3 Generating a flexible pattern

```
1: procedure generatePattern( instance (I, J, W, V, C) of the SCPKP)
2:     lowerBound ← ⌈C/∑_{j=1}^{J} W_j⌉
3:     k ← random integer in [lowerBound; n]
4:     P ← I sets {J}
5:     while card(P) ≤ k do
6:         i ← random integer in [1; I] such that |S_i(P)| < J
7:         j ← S_i(P)(1)
8:         S_i(P) ← S_i(P) ∪ {j − 1}
9:     end while
10:    while (ii) and (iii) not verified for P do
11:        X ← set in χ(P) for permutation π_id
12:        x_ij ∉ X ← variable chosen randomly such that x_{ij+1} ∈ X or j = J
13:        if no such x_ij remains then
14:            go to line 26
15:        end if
16:        if j = J and !verifyCondition(P, i, j) then          ▷ (see Algorithm 2)
17:            S_i(P) = S_i(P) ∪ {J}
18:        end if
19:        if j < J and !verifyCondition(P, i, j) then          ▷ (see Algorithm 2)
20:            S_i(P) = S_i(P) ∪ {j} \ {j + 1}
21:        end if
22:    end while
23:    if computeRank(P)=k then                                 ▷ (see Algorithm 1)
24:        return P
25:    else
26:        pattern discarded
27:    end if
28: end procedure
```

Property 20

Algorithm 3 only returns a pattern if it is a flexible pattern.

Proof : As $\mathcal{P}$ is discarded if $rank(\mathcal{P}) \neq k$, it ensures the rank of $\mathcal{P}$ to be exactly k. Similarly, pattern $\mathcal{P}$ verifies conditions (ii) and (iii). Indeed, condition (ii) is verified for every $i \in \{1,\dots,I\}$ and (iii) is verified for every $i \in \{1,\dots,I\}$ and $j \in S_i(\mathcal{P})$. Also, if a condition (ii) is verified for a given i, it is still verified if $\mathcal{P}$ is modified during the core of **Algorithm 3**. Similarly, if condition (iii) is verified for a given i and $j \in S_i(\mathcal{P})$, it is still verified if $\mathcal{P}$ is modified during the core of **Algorithm 3**. Indeed, as either one element is added, or an index j' is replaced by $j' - 1$ in a set $S_{i'}(\mathcal{P})$, then the same k-intersection satisfying (ii) or (iii) can be found, or a k-intersection with lighter total set weights. As $\mathcal{P}$ is initialized with one element per set, and the size of a set cannot decrease (when an index i is removed, it is replaced by $i - 1$), condition (i) is verified. Therefore, the **Algorithm 3** returns the pattern only if it verifies (i), (ii) and (iii) ■

Note however that pattern $\mathcal{P}$ obtained with **Algorithm 3** does not necessarily satisfy the condition of **Theorem 8**. Indeed, the algorithm does not restrict the generated pattern to have a set of cardinality 1. This means that the returned pattern is not necessarily a pattern-facet. However, the pattern being a flexible pattern ensures the associated inequality to have a known lower bound on the dimension of the associated face as proven in **Theorem 7**, thus leading to relatively strong inequalities.

Note that generatePattern$((I, J, W, V, C))$ only needs the instance, more precisely the knapsack bound and the item weight from the instance, hence it can be only used in pre-processing. Indeed, only the constraints are needed to compute a pattern verifying (i), (ii) and (iii). Also, **Algorithm 3** returns at most a single pattern, which may not be enough to make a large difference in a B&C framework. Consequently, we call this algorithm multiple times as the pre-processing step of the two-phase scheme, in order to generate various patterns. Each call is independent from the previous ones, meaning that multiple calls can be done in parallel on different threads. The set of patterns obtained with this algorithm is then used within a B&C framework, with the separation algorithm described in the following section.

4.6.3 Separation algorithm for the Symmetric-weight Chain Precedence Knapsack Problem

For the separation algorithm, we have two pieces of information: the fractional point $\widetilde{X}$ and the set of generated patterns. Also, recall that, for a given pattern, one can obtain the permutation maximizing the violation of the pattern inequality in polynomial time. This can be done by solving an MMP with the Hungarian algorithm [61], of complexity $\mathcal{O}((|\mathcal{H}_1| + |\mathcal{H}_2|)^4)$. A more recent version [33] is of complexity $\mathcal{O}((|\mathcal{H}_1| + |\mathcal{H}_2|)^3)$. In the scope of the separation algorithm, $|\mathcal{H}_1| = |\mathcal{H}_2| = I$. Thus, the separation algorithm is polynomial for a pattern, as it is of complexity $\mathcal{O}((I)^3)$.

In order to obtain the most violated pattern inequality for a set of patterns, the MMP is solved for each pattern. As the number of patterns is not bounded, obtaining the most violated pattern inequality out of a set of patterns is in pseudo-polynomial time. Note also that there is no guarantee that all pattern-facets have been generated. In such a case, this separation algorithm is a heuristic.

4.6.4 Two-phase Branch & Cut scheme

The first phase is to use **Algorithm 3** multiple times to generate as many flexible patterns as possible. As it requires the capacity C, the number of groups I, the number of items per group J and weights W_j, it can be done as a pre-processing. Because of the polyhedral symmetries of the SCPKP, each flexible pattern encodes an exponential number of pattern inequalities. From **Theorem 7**, there is a lower bound on the dimensions of the faces defined by these inequalities, and they are facet-defining in the case of a flexible pattern with a set of cardinality 1.

The second phase is to use inequalities associated with generated patterns within a B&C framework. As the patterns have already been generated, the separation algorithm only requires to find the permutation π leading to the most violated inequality for each pattern. Such a permutation can be found for a given pattern by solving the MMP, with a polynomial time algorithm.

The interest of such an approach is that a same flexible pattern can yield multiple cuts in the second phase. The following example shows how a single pattern can be used to separate multiple cuts.

Example 21

Let (3, 3, [3, 4, 2], [[14, 2, 10],[15, 2, 3],[14, 2, 9]], 10) be an instance of the SCPKP. For this instance, pattern $\mathcal{P} = \{\{1\},\{3\},\{3\}\}$ *is a flexible-pattern of rank 1. When solving the LP relaxation, we obtain the following solution,* $\widetilde{X}_1 = [[1, 0.166, 0.166], [1, 0, 0], [1, 0, 0]]$. *By finding permutation of* $\mathcal{P}$ *maximizing the violation, we obtain* $\mathcal{X}_1 = \{x_{13}, x_{23}, x_{31}\}$, *and* $(pi(\mathcal{X}_1))$ *is* $x_{13} + x_{23} + x_{31} \leq 1$. *Clearly, fractional solution* $\widetilde{X}_1$ *violates* $(pi(\mathcal{X}_1))$. *With this new cut we obtain a second fractional solution,* $\widetilde{X}_2 = [[1, 0, 0], [1, 0, 0], [1, 0, 166, 0, 166]]$. *By finding the permutation of* $\mathcal{P}$ *maximizing the violation, we obtain* $\mathcal{X}_2 = \{x_{13}, x_{21}, x_{33}\}$, *and* $(pi(\mathcal{X}_2))$ *is* $x_{13} + x_{21} + x_{33} \leq 1$. *Clearly, fractional solution* $\widetilde{X}_2$ *violates* $(pi(\mathcal{X}_2))$.

4.7 Experimental results

Results are computed on a single thread of an Intel Core i7-9850H CPU @ 2.60GHz processor, with 12 CPUs of 12 cores, with Linux as operating system. All algorithms are developed with C++. Version 12.8 of CPLEX is used to solve model M_{SCPK}. Recall that UMIC inequalities, introduced in **Section 4.2**, have been adapted to the SCPKP. Hence, we compare in this section seven variants of the B&C algorithms featuring the pattern inequalities, the inequalities featured in CPLEX and the UMIC inequalities. Besides, we also compare all combinations of these sets of inequalities.

4.7.1 Instance description

One of the main result when comparing the solving time of various knapsack problem instances [83] is that a stronger correlation between weight and value tends to induce a large increase of the computational times. Instance generators for variants of the knapsack problem are at disposal [1] with various characteristics. However, none of them produce instances for the particular case of the SCPKP, i.e., with I groups of J elements, chain precedence constraints and symmetric weights W_j, $j \leq J$. Recall that we study the SCPKP due to its structure being the core of the 1-HUC^D problem. However, we do not include instances for the 1-HUC^D problem with the Hydro unit commitment Instance Generator [2] for two reasons. First, the prices value in an instance of the 1-HUC^D problem is very structured, which

[1] http://hjemmesider.diku.dk/~pisinger/codes.html, accessed 2023-06-20
[2] http://www.lix.polytechnique.fr/Labo/Dimitri.Thomopulos/libraries/HIG.html

is not the case of the SCPKP . Second, many constraints of the 1-HUC^D problem are not taken into account in the study of the SCPKP , hence it would be difficult to identify the impact of the presented inequalities and algorithms. Consequently, we generate our own random instances by keeping a strong enough correlation between weights and values. From a pool of hundred generated instances, sixty are retained. The selection criteria is for these instances to take at least 60 seconds to be solved by CPLEX, with all the default options of CPLEX enabled but the cuts. As such, there is enough room in terms of computational time and number of explored B&C nodes to see the impact of each of the family of inequalities considered. The sixteen retained instances are as follows, instances 1 to 20 with $I = 20$ $J = 5$; instances 21 to 40 with $I = 30$ $J = 5$; instances 41 to 60 with $I = 20$ $J = 10$. Note that these instances are in the case $I \geq J$, for which the SCPKP is proven to be NP-hard. The generation of instances is further detailed in **B.2** and the resulting instance set is available online [3].

It would be interesting to see how a correlation similar to that proposed in [83] can extend to the generated SCPKP instances. When considering instances from the difficult classes in [83] the correlation coefficient is above 0.98. In the case of the SCPKP, item (i, j) can only be selected if $(i, j-1)$ has been selected, due to chain precedence constraints. Hence, one can compute the correlation coefficients between the weights and values, but also the correlation coefficient between the cumulated weights and cumulated value, i.e., $\sum_{j=1}^{J} W_j$ and $\sum_{j=1}^{J} V_{ij}$ instead of W_j and V_{ij}.

Table 4.3 gives the correlation between the weights and values (cor), as well as the correlation between the cumulated weights and cumulated values (cor_C). The considered linear Bravais-Pearson correlation formula is as follows:

$$cor = \frac{cov(W, V)}{\sigma_W \cdot \sigma_V};$$

with $cov(W, V)$ the covariance, and σ_W (resp. σ_V) the standard deviation of the weights (resp. values).

Interestingly, for the SCPKP instances, metric cor often takes a low or even negative value, whereas metric cor_C is always over 0.70. This indicates that cor_C is a better metric to evaluate the hardness of SCPKP instances. It also appears that metric cor tends to have a higher value for instances 41 to 60 than for the other instances. As these instances are with $J = 10$ items per group, it could mean that for instances with high J, the value of cor and cor_C become correlated. Note that this remark does not apply for the number of groups I, as there is no noticeable difference between instances 1 to 20 and 21 to 40. Note that values in **Table 4.3** are rounded to the nearest hundredth and value 1.00 does not necessarily mean a perfect correlation.

4.7.2 Pattern generation

To generate a set of patterns, we repeatedly generate patterns one by one via the pattern generation process depicted in **Algorithm 3**. This process is random-based, hence it can produce multiple times the same pattern. In this case, we only consider one copy of each pattern. Although the process is

[3] https://github.com/Eegann/SCPKinstances, accessed 2023-07-21

inst	cor	cor_C	inst	cor	cor_C	inst	cor	cor_C
1	0.82	0.85	21	−0.3	0.97	41	0.57	1.00
2	−0.67	0.94	22	0.29	0.98	42	0.56	0.98
3	0.36	0.98	23	0.18	0.99	43	0.68	0.99
4	0.51	0.99	24	0.69	0.99	44	0.59	0.99
5	0.7	1.00	25	0.42	0.99	45	0.67	0.98
6	0.03	0.96	26	0.65	0.99	46	0.37	0.99
7	0.72	0.99	27	0.73	0.98	47	0.35	0.99
8	0.81	0.97	28	0.87	1.00	48	0.36	0.99
9	0.33	0.99	29	0.32	1.00	49	0.52	0.99
10	0.65	1.00	30	−0.26	0.99	50	0.14	0.99
11	0.59	0.99	31	−0.8	0.99	51	0.61	0.99
12	0.97	0.99	32	0.69	0.99	52	0.48	0.99
13	0.0	0.99	33	0.44	0.98	53	0.2	0.99
14	0.49	0.97	34	0.48	0.92	54	0.44	1.00
15	0.61	0.97	35	−0.06	0.94	55	0.2	0.99
16	0.89	1.00	36	0.45	0.85	56	0.2	0.98
17	0.74	0.99	37	0.54	0.71	57	0.87	0.98
18	−0.03	0.99	38	0.03	0.98	58	0.54	1.00
19	0.82	0.99	39	0.58	0.98	59	0.9	0.98
20	−0.44	0.99	40	0.23	0.98	60	0.51	0.77

Table 4.3: Correlation coefficients between the value and weight for all instances

random-based, we only generate a single set of patterns for each instance. Indeed, preliminary results have shown that, for a given instance, generating multiple sets of patterns yields very similar results, both in terms of generated patterns, and B&C results to solve the SCPKP. Even if the pattern generation procedure could be parallelized, we use a single thread.

The following metrics are used in **Table 4.4** to compare the pattern generations for each instance:

- #iter: number of iterations of the generation process
- #find: number of iterations where a flexible pattern is found
- #patt: number of different flexible patterns found
- %facet: proportion of patterns guaranteed to be pattern-facet, i.e., with a set of cardinality 1

Firstly, one can see that the generation becomes slower on larger instances, thus with the same time limit, fewer iterations can be performed. More precisely, there are thousands of iterations for instances 1 to 20, around a thousand for instances 21 to 40, and hundreds for instances 41 to 60. Consequently, fewer patterns are found for larger instances. Note that there are many instances for which most iterations of the pattern generation procedure fail to produce a flexible pattern in a majority of cases. Indeed, for some instances, the procedure produces a flexible pattern during less than 10% of the iterations, see for example instances 4, 15, 22, 36, 44, 50. Generally speaking, the larger the instances, the higher the proportion of failed iterations.

Moreover, when #find is large, most of the flexible patterns are generated multiple times. Indeed, roughly speaking, the ratio between #patt and #find is smaller when #find is large. This is due to the procedure being random-based, hence when #find is large, there is a higher chance to generate duplicates.

inst	#iter	#find	#patt	%facet	inst	#iter	#find	#patt	%facet	inst	#iter	#find	#patt	%facet
1	5011	3245	244	4.9	21	1123	517	106	3.8	41	371	31	18	16.7
2	7428	3870	45	15.6	22	977	21	9	88.9	42	346	17	11	36.4
3	7432	6594	53	9.4	23	1290	947	186	2.7	43	338	31	18	16.7
4	4070	385	78	19.2	24	851	69	34	14.7	44	339	10	9	44.4
5	4112	1125	207	3.9	25	815	93	50	20.0	45	299	12	8	37.5
6	3935	274	61	29.5	26	1001	332	140	9.3	46	484	97	53	7.5
7	4421	1428	199	5.5	27	990	200	43	11.6	47	470	126	82	4.9
8	3414	210	26	23.1	28	773	68	32	12.5	48	650	326	205	1.5
9	3603	281	50	26.0	29	859	305	144	5.6	49	864	616	182	1.1
10	3935	395	74	17.6	30	870	38	20	40.0	50	375	5	2	100.0
11	4059	1078	195	7.2	31	859	172	72	12.5	51	354	50	40	7.5
12	4649	2008	296	3.4	32	855	34	11	72.7	52	336	38	22	18.2
13	4444	644	125	14.4	33	1208	36	8	100.0	53	374	6	4	50.0
14	5937	2484	96	4.2	34	795	98	44	22.7	54	727	344	200	1.0
15	3501	250	40	20.0	35	790	82	38	23.7	55	385	80	55	5.5
16	9023	308	11	100.0	36	880	25	8	62.5	56	863	587	187	1.1
17	4112	1393	166	4.2	37	960	169	52	11.5	57	297	2	2	0.0
18	4294	192	33	45.5	38	1027	22	7	57.1	58	363	6	6	50.0
19	5125	1787	152	7.9	39	812	177	94	4.3	59	366	43	34	5.9
20	3869	287	57	28.1	40	804	29	20	50.0	60	517	102	46	4.3

Table 4.4: Experimental results relative to pattern generation (first phase)

4.7.3 Separation of inequalities

Separation of the UMIC inequalities For UMIC inequalities, we use the uplifting algorithm and implemented the separation process as described in [37]. This algorithm first generates a smaller instance, with a *contracted graph*, based on the fractional solution. On the smaller instance, a MIC is found for each variable. For each MIC, computing coefficients β_{ij} for a given $(i,j) \in U_r$ to optimality is NP-hard. Hence, the linear relaxation is considered instead. For a given $(i,j) \in U_r$, if β_{ij} is fractional, then it is rounded up to enforce the inequality. This process yields UMIC inequalities for the instance relative to the contracted graph, which is then translated back to the original instance. The algorithm returns only the three most violated UMIC inequalities.

We implemented the separation with the following features as described in [37]. In particular, the separation is only enabled at the root node. As long as a violated inequality is found, the LP relaxation is recomputed and the separation process is repeated. The separation procedure is disabled once the number of UMIC inequalities added reaches ten times the number of inequalities of the model.

Separation of the pattern inequalities The separation of the pattern inequalities is done within a user cut callback from CPLEX. For a given pattern set, it reduces to solve an MMP for each of its patterns (see **Section 4.5**), hence creating one inequality per pattern, and retains the most violated inequality. Preliminary results show that this strategy is, on average, more efficient than retaining the first violated inequality, or all the violated inequalities.

This process is repeated at each node, as long as a violated inequality is found. This means that the separation of the pattern inequalities is made at least once per node. Recall that an MMP is solved for each pattern, and from **Table 4.4** the number of patterns can reach hundreds. Preliminary results show that using the separation of the pattern inequalities at each node takes up to 90% of the total computational time, and the larger the tree, the fewer the cuts. Besides, preliminary results also show that few patterns yield the majority of the added cuts. Hence, we propose the following scheme, which takes advantage of the fact that we have a set of patterns. For each pattern, we compute the average violation of the added cuts, which is 0.0 when no cut has been added. The set of patterns is kept sorted by decreasing value of this average violation. The idea is to divide the size of the set by two, keeping only the first half of the set, such that the set is empty at 10,000 nodes. More precisely, the number of times the set must be divided is $\log_2$ of the number of patterns. Hence, one can divide 10,000 by this number to know at which number of nodes of the B&C the set is divided. Besides, we also limit the number of added cuts to 100. The separation of the patterns is disabled once either the set of patterns is empty, or the number of added cuts has reached 100.

Note that the violation of a cut does not have an upper bound of 1 despite variables being binary, as illustrated by Example 22.

Example 22

Let (2, 2, [10, 5], V, 25) be an instance of the SCPKP. One can obtain the fractional solution $\tilde{x}_{11} = 1$, $\tilde{x}_{12} = 1$, $\tilde{x}_{21} = 0.66$, $\tilde{x}_{22} = 0.66$. For this instance, inequality $\tilde{x}_{12} + \tilde{x}_{21} + \tilde{x}_{22} \leq 1$ is valid and has a violation value of 1.32 when cutting the fractional solution.

Separation of UMIC and pattern inequalities. When both UMIC and pattern inequalities are considered, we first generate pattern inequalities at the root node. Once no more pattern inequalities are violated at the root node, we then generate UMIC inequalities as previously defined. When no more UMIC inequalities are violated, we resume the B&C algorithm and separate the pattern inequalities as described previously.

4.7.4 Solving the Symmetric-weight Chain Precedence Knapsack Problem

The B&C framework is limited to a single thread, and a maximum of 3600 seconds of computational time. For the following, we define *default CPLEX*, CPLEX with all default options enabled, and *no-cut*

CPLEX, which is default CPLEX with cuts disabled. Seven experimental variants of CPLEX's B&C are considered in order to evaluate the pattern inequalities:

- Cplx: default CPLEX.
- Umic: no-cut CPLEX with UMIC inequalities separation.
- Psep: no-cut CPLEX with pattern inequalities separation.
- Cplx+Umic: default Cplex with UMIC inequalities separation.
- Cplx+Psep: default CPLEX with pattern inequalities separation.
- Psep+Umic: no-cut CPLEX with pattern inequalities separation and UMIC inequalities separation.
- All: default CPLEX with pattern inequalities separation and UMIC inequalities separation.

In order to compare these variants, we first introduce **Figure 4.5**, representing the number of instances solved for each variant with respect to the computational time. Note that, for readability purposes, the scale is linear between 0 and 500 seconds, and becomes logarithmic between 500 and 3600. **Figure 4.6** is similar but with respect to the number of nodes. For this figure, the scale is linear between 0 and 2 million nodes, and then becomes linear between 2 million and 60 million nodes.

For more detailed results, we also introduce **Table 4.5**, which contains the numerical results for a representative subset of the instances. The complete tables are in **Appendix B.3**. In these tables, we use the following metrics :

- *inst*: the instance number.
- *variant*: experimental variant of CPLEX's B&C considered.
- *C cuts*: number of added CPLEX cuts.
- *U cuts*: number of added UMIC cuts.
- *P cuts*: number of added pattern cuts.
- *P cuts vio*: average violation value of the added pattern cuts.
- *r-value*: linear relaxation value at the root node.
- *r-gap*: gap between the linear relaxation value at the root node and the optimal solution, if optimality is proven by at least one variant.
- *user-time*: proportion of the computational time dedicated to the separation of pattern and UMIC inequalities.
- *#nodes*: number of nodes explored.
- *time/gap*: total computational time in seconds when the instance is solved; gap between the upper and lower bound provided by CPLEX at the time limit when the instance is not solved.

For further details, the types of cuts added by CPLEX are in **Appendix B.4**.

General results In **Figure 4.5**, we first notice that, variant Psep outperforms Umic and Cplx. Besides, variants Cplx+Psep and All are the most efficient for instances solved within 100 seconds. For harder to solve instances, variant Psep+Umic is the most effective, closely followed by variant Psep. These results

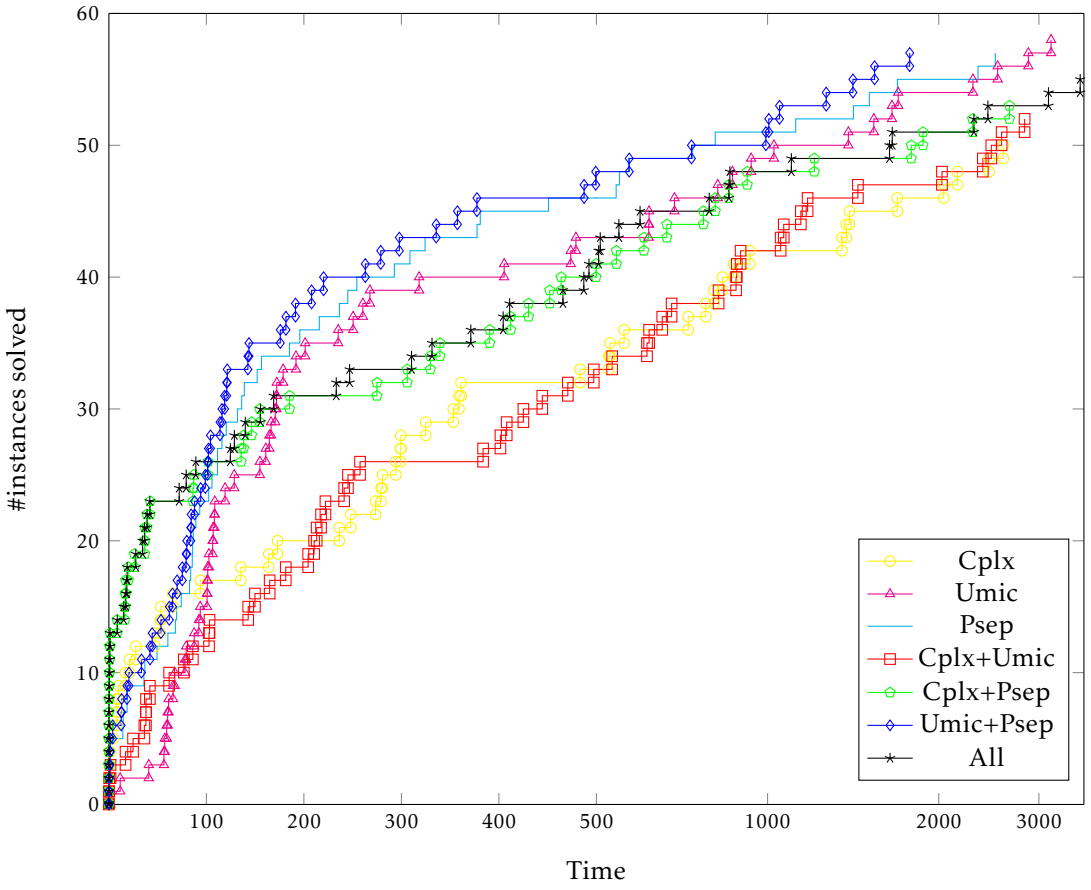


Figure 4.5: Number of instances solved with respect to the computational time

show that not only Psep is an interesting variant on its own, but also is featured in all most efficient combinations.

When it comes to **Figure 4.6**, similar conclusions can be drawn for the most efficient variants. Indeed, variants Cplex+Psep and All are the most efficient variants for instances solved with less then 10 million nodes, and then Psep+Umic becomes the most efficient. Hence, using Psep reduces the number of nodes explored.

The efficiency of using Psep is shown in the results as there are many instances where Psep clearly reduces the computational time. Moreover, there are many cases where variants featuring Psep solve the instance at the root node. We enumerate these instances, and emphasize in bold when only a variant with Psep can solve the SCPKP at the root node: 1, 2, **3**, **8**, **21**, **27**, 29, **32**, 33, **34**, **46**, 49, 59. There are more than 10% of the instances that are solved at the root node only by a variant featuring Psep. Note also that these improvements are sometimes due to very few inequalities. Less than 7 inequalities are added for instance 3, less than 14 for instance 27, and only 1 for instance 32.

Note however that there are instances where Psep is outperformed by Cplex or Umic. For example, Umic outperforms Psep for instance 5 and Cplex outperforms Psep for instance 14. This suggests that

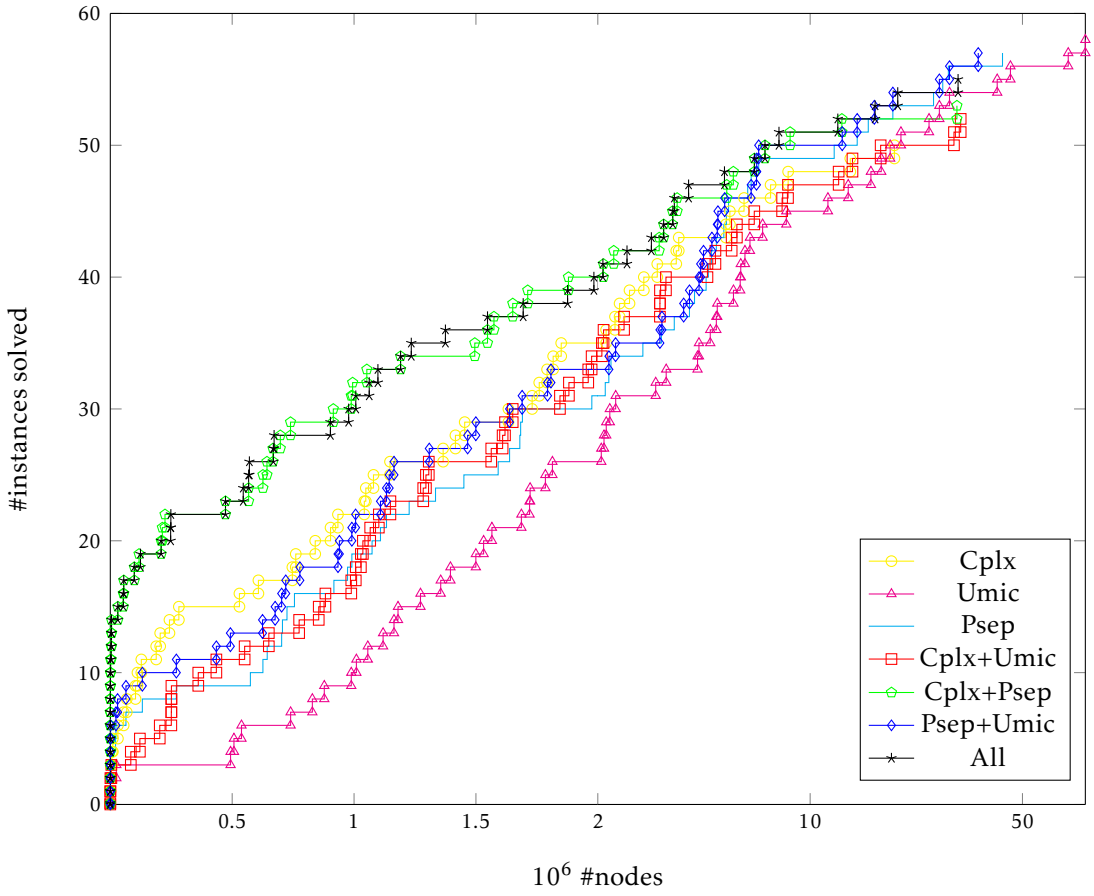


Figure 4.6: Number of instances solved with respect to the number of nodes

even if binary cuts are added by variants Psep and Umic, they do not add the exact same ones. For instance, recall that we showed in **Counter-example 8** that there are binary inequalities of the SCPKP captured by pattern inequalities, but not by the inequalities of the literature. As it is not known if Cplx adds only binary inequalities, it is not possible to deduce that binary inequalities from Cplx are different than the ones of Umic and Psep. However we can still suppose that all variants produce different inequalities, as in **Figure 4.6** and **Figure 4.5**, variants combining Psep and either Umic or Cplx outperform the other variants.

We also note two interesting results when it comes to the quality of the dual bound at the root node. First, the r-value and r-gap are usually the highest for variant Psep. This suggests that, for such cases, very few to no cuts are added at the root node by Psep, and this variant requires some branching in order to produce efficient cuts. Second, the metric %facet from **Table 4.4** seems to be related to neither the computational time nor the number of pattern cuts added. Indeed, on the one hand, metric %facet is below 10% for instance 3, yet it is solved at the root node by Psep. On the other hand, metric %facet is 20% for instance 25, yet none of the pattern inequalities has been added by Psep, and the computational

inst	variant	C cuts	U cuts	P cuts	P cuts vio	r value	r-gap	user time	#nodes	time/gap
	Cplx	405	-	-	-	5234.55	9.41%	-	1849854	173.0
	Umic	-	426	-	-	5214.77	8.99%	8.3%	2095765	160.9
	Psep	-	-	7	0.9	4829.68	0.95%	35.5%	0	0.0
3	Cplx+Umic	58	315	-	-	5200.3	8.69%	6.7%	2088692	164.9
	Cplx+Psep	5	-	3	0.76	4795.72	0.24%	17.9%	0	0.0
	Psep+Umic	-	0	7	0.9	4829.68	0.95%	33.1%	0	0.0
	All	5	0	3	0.76	4795.72	0.24%	15.9%	0	0.0
	Cplx	405	-	-	-	14288.9	1.10%	-	1816805	280.6
	Umic	-	27	-	-	14302.5	1.20%	1.1%	1180341	56.5
	Psep	-	-	100	0.11	14291.1	1.12%	2.7%	1226118	93.2
5	Cplx+Umic	231	33	-	-	14283.4	1.06%	0.7%	1066178	142.9
	Cplx+Psep	143	-	100	0.11	14291.0	1.12%	1.4%	988415	146.4
	Psep+Umic	-	7	100	0.11	14290.5	1.12%	3.5%	1144164	79.9
	All	139	6	100	0.12	14290.5	1.12%	1.6%	1062315	139.9
	Cplx	305	-	-	-	19621.1	2.02%	-	22632994	0.45%
	Umic	-	438	-	-	19674.1	2.29%	0.9%	19907454	1697.2
	Psep	-	-	100	0.29	19544.5	1.62%	0.5%	1638025	136.2
8	Cplx+Umic	178	15	-	-	19618.0	2.00%	0.6%	31306241	2831.5
	Cplx+Psep	3	-	2	0.74	19232.8	0.00%	17.8%	0	0.0
	Psep+Umic	-	0	100	0.29	19544.5	1.62%	0.7%	1638025	121.4
	All	3	0	2	0.74	19232.8	0.00%	15.4%	0	0.0
	Cplx	210	-	-	-	2164.88	1.14%	-	127247	8.3
	Umic	-	188	-	-	2181.78	1.93%	2.3%	2131653	79.3
	Psep	-	-	100	0.25	2185.22	2.09%	1.5%	2209549	86.5
14	Cplx+Umic	156	45	-	-	2164.22	1.11%	6.6%	202893	17.1
	Cplx+Psep	83	-	100	0.46	2166.64	1.23%	3.2%	224382	17.2
	Psep+Umic	-	188	100	0.24	2181.78	1.93%	2.8%	1807699	79.2
	All	57	42	100	0.55	2164.22	1.11%	9.1%	247563	17.4
	Cplx	605	-	-	-	6672.83	1.14%	-	10830484	0.08%
	Umic	-	482	-	-	6675.75	1.18%	1.2%	18347628	1387.1
	Psep	-	-	100	0.21	6643.91	0.70%	57.6%	1381	1.0
21	Cplx+Umic	354	25	-	-	6672.14	1.13%	0.4%	17067043	2390.8
	Cplx+Psep	4	-	2	0.48	6597.65	0.00%	43.2%	0	0.0
	Psep+Umic	-	0	100	0.21	6643.91	0.70%	71.2%	1381	1.2
	All	4	0	2	0.48	6597.65	0.00%	41.9%	0	0.0
	Cplx	605	-	-	-	28438.1	0.66%	-	12354812	0.16%
	Umic	-	6	-	-	28455.7	0.72%	0.9%	28764074	1537.8
	Psep	-	-	0	nan	28456.6	0.73%	1.8%	28518001	1509.9
25	Cplx+Umic	360	9	-	-	28438.1	0.66%	0.2%	14436657	0.2 %
	Cplx+Psep	363	-	0	nan	28438.1	0.66%	0.6%	14007958	0.2 %
	Psep+Umic	-	6	0	nan	28455.7	0.72%	2.1%	28764074	1412.8
	All	361	6	0	nan	28438.1	0.66%	0.7%	14962100	0.19%
	Cplx	605	-	-	-	8548.64	1.17%	-	2834093	778.5
	Umic	-	390	-	-	8544.39	1.12%	2.0%	6337030	618.0
	Psep	-	-	14	0.43	8453.1	0.04%	59.8%	3	0.1
27	Cplx+Umic	228	218	-	-	8544.35	1.12%	0.8%	4590923	820.1
	Cplx+Psep	2	-	5	0.41	8454.58	0.06%	34.3%	0	0.0
	Psep+Umic	-	0	14	0.43	8453.1	0.04%	89.6%	3	0.2
	All	2	0	5	0.41	8454.58	0.06%	34.6%	0	0.0
	Cplx	415	-	-	-	39078.4	0.27%	-	7410282	1375.8
	Umic	-	6	-	-	39171.6	0.51%	1.1%	17142128	685.9
	Psep	-	-	1	0.85	39079.8	0.28%	4.9%	4162387	120.4
32	Cplx+Umic	359	12	-	-	39077.1	0.27%	0.3%	6547609	1054.3
	Cplx+Psep	6	-	1	0.85	38998.1	0.07%	16.5%	0	0.0
	Psep+Umic	-	12	1	0.85	39077.3	0.27%	5.6%	4455866	114.0
	All	8	8	1	0.85	38998.1	0.07%	82.5%	0	0.1
	Cplx	820	-	-	-	29884.3	0.45%	-	903545	559.2
	Umic	-	9	-	-	29902.1	0.51%	0.6%	739761	57.3
	Psep	-	-	0	nan	29911.9	0.55%	6.0%	706313	60.5
53	Cplx+Umic	540	9	-	-	29886.0	0.46%	0.1%	774691	401.6
	Cplx+Psep	543	-	0	nan	29884.3	0.45%	1.0%	697532	390.6
	Psep+Umic	-	9	0	nan	29902.1	0.51%	6.9%	719822	53.4
	All	540	9	0	nan	29878.2	0.43%	1.4%	570611	310.4

Table 4.5: Experimental results relative to a subset of SCPCK instances (second phase)

time for Psep is above 1500 seconds. For instances where the pattern cuts lead to a short computational time, it could mean that many of the flexible patterns without a set of cardinality 1 are also pattern-facet.

Results per type of instances. One limitation of Psep that stands out is that sometimes, no pattern inequality is violated. This is due to the limited set of patterns. Besides, when no pattern inequality is found, there are also very few UMIC inequalities. See for example instances 15, 24, 25 of 43, 50, 51, 53, 58, where about ten UMIC inequalities where added, even if the maximal number of UMIC inequalities is much larger. This suggests that, for these instances, another set of inequalities, such as integer inequalities, may be more relevant. Note also that the instances where no pattern inequalities where found tend to be large instances. Indeed, one instance with $I = 20$ and $J = 5$, two with $I = 30$ and $J = 5$ and five with $I = 20$ and $J = 10$. This can be due to the pre-processing time limit being the exact same, independently of the size of the instance.

In order to further compare the variants, we define the three following metrics:

- *#best*: the number of instances where a variant features the smallest computational time, or the smallest gap if no variant solves the instance to optimality.
- *#solved*: the number of instances solved by a variant
- *avg-time*: the average time for instances solved by a variant

Note that sometimes, the computational time of two variants are very similar for a given instance. In such a case, distinguishing a single best variant may be unfair, hence we will consider multiple best variants when their computational times are close. For a given instance, let $time^*$ be the computational time of the quickest variant. Any variant is considered to be the best if their computational time is below $1.1 \cdot time^*$ or $time^* + 1$. Let gap^* be the smallest gap for instances that are solved by none of the variants. As for the computational time, any variant is the best variant if its gap is below $1.1 \cdot gap^*$ or $gap^* + 0.1$.

Based on these metrics, we define two rankings. Ranking R_1 is solely based on metric #best. Note that two variants can have the same ranking for R_1. Ranking R_2 is based on #solved, and on avg-time to break a tie. **Table 4.6** to **4.8** show respectively for each set of instance, the value of each metric and the ranking for R_1 and R_2 of each variant.

variant	#best	#solved	avg-time-solved	R_1	R_2
Cplx	2	18	499.10	6	7
Umic	5	19	426.63	5	5
Psep	6	19	185.23	2	2
Cplx+Umic	1	19	578.63	7	6
Cplx+Psep	6	19	322.06	2	4
Psep+Umic	9	19	170.03	1	1
All	6	19	295.96	2	3

Table 4.6: Metrics for each variant on the set of instances with $I = 20$ and $J = 5$

variant	#best	#solved	avg-time-solved	R_1	R_2
Cplx	6	14	446.34	3	7
Umic	5	19	725.79	5	1
Psep	3	18	513.20	6	3
Cplx+Umic	3	15	571.03	6	6
Cplx+Psep	7	16	310.85	2	5
Psep+Umic	5	18	374.13	4	2
All	8	17	460.43	1	4

Table 4.7: Metrics for each variant on the set of instances with $I = 30$ and $J = 5$

variant	#best	#solved	avg-time-solved	R_1	R_2
Cplx	1	18	630.96	6	7
Umic	8	20	361.62	2	3
Psep	7	20	348.45	3	2
Cplx+Umic	1	18	628.51	6	6
Cplx+Psep	4	18	505.75	4	5
Psep+Umic	11	20	328.75	1	1
All	4	19	631.17	4	4

Table 4.8: Metrics for each variant on the set of instances with $I = 20$ and $J = 10$

These tables emphasize the improvement of the resolution of the SCPKP when using pattern inequalities. Indeed, all four variants featuring pattern inequalities (Psep, Cplex+Psep, Psep+Umic, All) always contain the best variant for both rankings, the only exception is Umic being the best variant with respect to R_2 in **Table 4.7**. Besides, variant Cplx always occupy a worse position than Cplx+Psep with respect to both R_1 and R_2, and Umic always occupy a worse position than Psep+Umic for R_1, and for two out of the three tables for R_2.

Tables 4.6 to **4.8** also show that increasing the number of groups I makes the instances drastically harder, while increasing the number of items J makes the instances slightly harder. Indeed, the number of instances solved #solved is significantly smaller in **Table 4.7** than in the two other tables. Despite #solved being similar in **Table 4.6** and **4.8**, the average computational time avg-time-solved is clearly larger in **Table 4.8**. Hence, even if instances with $I = 30$ and $J = 5$ have fewer variables than instances with $I = 20$ and $J = 10$, the former are more difficult than the latter. This could be explained by the fact that with more groups I, the polyhedral symmetries as defined in **Definition 4** are further increased, which is not necessarily the case when increasing the number of items J.

4.8 Conclusion

In this chapter, the Symmetric-weight Chain Precedence Knapsack Problem (SCPKP) is considered as a new variant of the Knapsack Problem, and as a special case of the Precedence Knapsack Problem

(PKP). As such, we adapt uplifted minimal induced cover inequalities defined for the PKP to the case of the SCPKP. The SCPKP features polyhedral symmetries, which generalize the classical symmetries. Two main contributions are proposed, namely the polyhedral study and the two-phase Branch & Cut scheme, revolving around the patterns introduced to handle polyhedral symmetries. We derived pattern inequalities as a new class of valid inequalities embedding symmetries with respect to the groups. Necessary facet-defining conditions are defined for these inequalities. These conditions are also sufficient in the case of a pattern with a set of cardinality 1. The separation problem for pattern inequalities is defined as well as proven NP-hard for a family of patterns. An algorithm is presented to generate a set of patterns verifying such conditions as pre-processing and first phase of the scheme. A separation algorithm based on the generated patterns is presented and reduces to match the patterns with the fractional point in order to produce a violated inequality. This algorithm is used within a Branch & Cut framework in the second phase of the scheme. Experimental results demonstrate the efficiency of the pattern inequalities and separation algorithms against CPLEX's inequalities and uplifted minimal induced cover inequalities. Results show that our two-phase Branch & Cut algorithm outperform a Branch & Cut algorithm featuring any of these two sets of inequalities. Besides, when combining multiple sets of inequalities, combinations featuring pattern inequalities yield the best results.

In the next chapter, we extend the results and algorithms designed for the SCPKP to the discretized 1-Hydro Unit Commitment problem. For such an extension, we also translate the results to the Inverted SCPKP, with a covering inequality rather than a knapsack inequality. As such, the resource windows of the 1-Hydro Unit Commitment problem can be taken into account when combining multiple SCPKPs and Inverted SCPKPs.

5 Polyhedral study for the discretized 1-Hydro Unit Commitment problem without ramping nor min-up/down constraints

Table of contents

In the previous chapter, we defined the Symmetric-weight Chain Precedence Knapsack Problem (SCPKP), for which we conducted a polyhedral study, and defined a specific two-phase Branch & Cut algorithm. The reason for studying this problem is that it features a subset of inequalities of the discretized 1-Hydro Unit Commitment problem without ramping nor min-up/down constraints (1-HUC^D problem). However, this subset does not capture the resource windows of the 1-HUC^D problem. As such, we aim at extending the results of the SCPKP to the 1-HUC^D problem, particularly to take into account these resource windows. For this purpose, we introduce the Inverted SCPKP (ISCPKP), so that each resource window of the 1-HUC^D problem corresponds to a pair of an SCPKP and an ISCPKP.

In this chapter, we present a polyhedral study of the 1-HUC^D problem, based on the polyhedral study of the SCPKP. In **Section 5.1**, we define the problems featured in this polyhedral study. In **Section 5.2**, we review polyhedral studies for problems related to the considered variant of the 1-HUC problem. In **Section 5.3**, we translate the results of the SCPKP to the ISCPKP. In **Section 5.4**, we show that one can represent the constraints of the 1-HUC^D problem with multiple (I)SCPKPs. In **Section 5.5**, we extend the polyhedral study to the considered 1-HUC^D problem. In **Section 5.6**, we generalize the two-phase Branch & Cut algorithm to the 1-HUC^D problem before concluding.

5.1 Problems definition

The 1-HUC problem variant considered in this section is the 1-HUC^D problem defined in **Section 3.3**. We define an instance for the 1-HUC^D problem as follows. For the purpose of the polyhedral study, we only need the constraints, hence we define an instance without specifying the value $\Psi_{t,i}$ of the operating points.

Definition 29 (*Instance of the 1-HUC^D problem*)

We denote (N, T, D, β^*, α^*) an instance of the 1-HUC^D problem, with N operating points, T time periods, D the water flow vector and β^* (resp. α^*) the lower bound (resp. upper bound) vector.

We also consider in this section the SCPKP, defined in **Section 4.1**.

In order to extend the polyhedral study of the SCPKP to the 1-HUC^D problem, we introduce the Inverted SCPKP (ISCPKP). The ISCPKP is an SCPKP with a covering inequality rather than a knapsack inequality. Hence, in the case of the ISCPKP capacity C is a minimal bound on the weight of the selected items. Solving the ISCPKP is to maximize the total value of the selected items, while the chain precedence constraints are satisfied, and the total weight of the selected items is greater than or equal to capacity C. Note that the ISCPKP is not trivial, as values $V_{ij} \in \mathbb{R}$, there can be items with $V_{ij} < 0$. We also introduce ISCPKP instances as follows

Definition 30 (*Instance of the ISCPKP*)

We denote (I, J, W, V, C) an instance of the ISCPKP, with I the number of groups, J the number of elements, W the weight vector, V the value matrix and C the covering capacity.

5.2 Related problems polytopes

As the aim of this chapter is to extend a polyhedral study of the SCPKP to the 1-HUC^D problem we review polyhedral studies related to the latter. More specifically, this extension concerns mainly the minimal and maximal bound of the cumulated flow (3.2.6) and (3.2.7) of $M_{op\text{-}D}$. We precised in **Chapter**

3 that constraints (3.2.6) and (3.2.7) of model $M_{op\text{-}D}$ can be seen as resource windows. As such, we also review polyhedral results for these types of constraints.

5.2.1 (Hydro) Unit Commitment polytopes

As stated previously, the 1-HUC is a non-linear problem from the power function arising in the objective function. In the survey [97], most of the reported work dedicated to the 1-HUC problem aims at proposing MILP models of high precision. These works usually involve a piecewise linear function which is out of the scope of this polyhedral study, but is one of the highlighted models in **Chapter 2**. More generally, very few works reported concern a (H)UC problem with operating points.

When it comes to polyhedral studies, most of them are for ramping constraints (3.1.9) to (3.1.10) or min-up/down constraints (3.1.11) to (3.1.13). This is because these constraints also appear in other production problems such as the UCP. In [63], the min-up/down polytope is studied, and a family of valid inequalities is introduced, namely the alternative up/down inequalities, which completely describe the convex hull of the min-up/down polytope. A branch & cut algorithm is described in order to implement this new set of inequalities. In [85] turn-on/off inequalities are introduced. These inequalities are shown to dominate the alternative up/down inequalities. Indeed, turn-on/off inequalities also provide a complete description of the min-up/down polyhedron, and there are fewer turn on/off inequalities than alternative on/off inequalities. This is why turn-on/off inequalities are the ones featured in $M_{op\text{-}DRM}$, being constraints (3.1.11)-(3.1.13). In [29] valid inequalities for the ramping polytope are defined along with conditions under which these inequalities are facet-defining. Besides, the special case of the ramping polytope with two time periods is considered, as the number of facet-defining inequalities is linear and does not need a separation algorithm. In [78] the min-up/down and ramping polytope is studied. Valid inequalities are defined, as well as necessary facet-defining conditions. For some special cases with three time periods or less, the inequalities introduced define the convex hull of the polytope. In [74] the authors describe a tight formulation of the UC taking into account the min-up/down constraints and generation limits. We do not consider generation limits in our model, as they can be expressed through min-up/down constraints when considering operating points.

In our study, we study the 1-HUC^D problem, focusing on the resource windows (3.2.6) and (3.2.7). The aforementioned polyhedral studies focus on ramping and min-up/down constraints. As the 1-HUC^D problem does not feature ramping nor min-up/down constraints, the reviewed polyhedral studies for the UCP do not apply to our work. Note however that our polyhedral study complements the literature, as the 1-HUC^{DRM} problem features both the ramping and min-up/down constraints, as well as resource windows.

5.2.2 Resource windows

Resource windows are a generalization of resource constraints, when the resource has both a lower and upper bound.

The survey [56] depictes various solution techniques to solve variants of the Resource Constrained Shortest Path Problem (RCSPP), including variants with window constraints. Recall that the RCSPP is a special case of the Shortest Path Problem with Resource Windows (RWSPP), where only an upper bound on the resource exists. The techniques described in this survey are dynamic programming, Lagrangian decomposition, constraint programming and heuristics. No polyhedral results are presented.

In the survey [102], a state-of-the-art review of different shortest path variants is described. One of the variants presented is the RWSPP, for which it is indicated that there are few studies. Different works have been cited for the RWSPP, but none of them is a polyhedral study.

In [6], the polytope of the Asymmetric Traveling Salesman problem with Time Windows is studied. Three families of inequalities are presented. The first two families contain enhanced tournament constraints and lifted tournament constraints. The third family contains enhanced predecessor-successor inequalities [8]. The way these inequalities have been enhanced is by considering non-feasible paths due to the time window constraints. However, as stated in **Section 1.5**, time windows are less restrictive than resource windows due to the allowed waiting time. Hence these works do not extend to the 1-HUC^D problem.

5.3 The Inverted Symmetric-weight Chain Precedence Knapsack Problem

In this section, we aim to translate results of the SCPKP to the ISCPKP by reformulating the latter.

5.3.1 The Inverted Symmetric-weight Chain Precedence Knapsack Problem as a Symmetric-weight Chain Precedence Knapsack Problem

First, we define M_{ISCPK} a model for the ISCPKP:

$$\max \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij}\cdot y_{ij}$$

$$s.c. \sum_{i=1}^{I}\sum_{j=1}^{J} W_j \cdot y_{ij} \geq C \tag{5.3.1}$$

$$y_{ij} \geq y_{ij+1} \qquad \forall y_{ij} \in \mathcal{V}, j < J \tag{5.3.2}$$

$$y_{ij} \in \{0,1\} \qquad y_{ij} \in \mathcal{V}$$

The idea of the following is to reformulate the ISCPKP so that it has the same constraints as the SCPKP. For this purpose, we perform a variable substitution, in order to reformulate the ISCPKP as an

SCPKP. Let $\overline{x}_{ij} \in \overline{\mathcal{V}}$ be the binary variable such that $\overline{x}_{ij} = 1$ if item (i, j) is not selected in the solution. Clearly, $x_{ij} = 1 - \overline{x}_{ij}$ and $\overline{x}_{ij} = 1 - x_{ij}$ due to their binary nature. The model of the ISCPKP with $\overline{x}_{ij}$ is the following.

$$
\begin{aligned}
\max & \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij} \cdot (1 - \overline{x}_{ij}) \\
s.c. & \sum_{i=1}^{I}\sum_{j=1}^{J} W_j \cdot (1 - \overline{x}_{ij}) \geq C \\
& 1 - \overline{x}_{ij} \geq 1 - \overline{x}_{ij+1} && \forall \overline{x}_{ij} \in \overline{\mathcal{V}}, j < J \\
& 1 - \overline{x}_{ij} \in \{0, 1\} && \overline{x}_{ij} \in \overline{\mathcal{V}}
\end{aligned}
$$

which can be written as

$$
\begin{aligned}
\max & \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij} - \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij} \cdot \overline{x}_{ij} \\
s.c. & \sum_{i=1}^{I}\sum_{j=1}^{J} W_j \cdot \overline{x}_{ij} \leq I \cdot \sum_{j=1}^{J} W_j - C \\
& \overline{x}_{ij} \leq \overline{x}_{ij+1} && \forall \overline{x}_{ij} \in \overline{\mathcal{V}}, j < J \\
& \overline{x}_{ij} \in \{0, 1\} && \overline{x}_{ij} \in \overline{\mathcal{V}}
\end{aligned}
$$

The difference between this model and M_{SCPK} is the order constraints which are reversed. However, as the order constraints are the only ones depending on the index j, one can consider the renumbering $(j) = J - j$ to obtain a new model $M_{R\text{-}ISCPK}$ with the exact same constraints as M_{SCPK}:

$$
\begin{aligned}
\max & \sum_{i=1}^{I}\sum_{(j)=1}^{J} V_{i(j)} - \sum_{i=1}^{I}\sum_{(j)=1}^{J} V_{i(j)} \cdot \overline{x}_{i(j)} \\
s.c. & \sum_{i=1}^{I}\sum_{(j)=1}^{J} W_{(j)} \cdot \overline{x}_{i(j)} \leq I \cdot \sum_{(j)=1}^{J} W_{(j)} - C \\
& \overline{x}_{i(j)} \geq \overline{x}_{i(j+1)} && \forall \overline{x}_{i(j)} \in \overline{\mathcal{V}}, (j) < J \\
& \overline{x}_{i(j)} \in \{0, 1\} && \overline{x}_{i(j)} \in \overline{\mathcal{V}}
\end{aligned}
$$

As only the objective function differs between this model $M_{R\text{-}ISCPK}$ and M_{SCPK} the same polyhedral results apply for both.

5.3.2 Translating polyhedral results to the Inverted Symmetric-weight Chain Precedence Knapsack Problem

As the constraints of the ISCPKP can be reformulated as constraints of the SCPKP, we can extend all polyhedral results. This means that the full dimensional condition (fd) applies directly to model $M_{R\text{-}ISCPK}$. Condition (fd) can then be translated for model M_{ISCPK} as follows:

Definition 31 (*ISCPKP full dimensional condition (ifd)*)

An ISCPKP verifies (ifd) if for any item (i,j) there is at least one feasible solution where (i,j) is not selected.

Property 21

Any instance of the ISCPKP that does not verify (ifd) can be transformed into an instance of the ISCPKP that verifies (ifd), with the exact same solutions.

The proof for **Property 21** is similar to the one of condition (fd) in **Property 3**.

Necessary facet-defining conditions (i) , (ii) and (iii) also apply for model $M_{R\text{-}ISCPK}$. We show that one can transform a pattern for the ISCPKP modeled with $M_{R\text{-}ISCPK}$ into a pattern for M_{ISCPK}. Let $\overline{\mathcal{P}}$ be a pattern for the ISCPKP modeled with $M_{R\text{-}ISCPK}$. For a given $\overline{\mathcal{X}} \in \chi(\overline{\mathcal{P}})$, the inequality associated to $\overline{\mathcal{P}}$ is:

$$\sum_{\overline{x}_{i(j)} \in \overline{\mathcal{X}}} \overline{x}_{i(j)} \leq rank(\overline{\mathcal{P}})$$

If we reverse the variable change $x_{ij} = 1 - \overline{x}_{ij}$, we obtain the *inverted pattern inequalities* for the ISCPKP:

Definition 32 (*Inverted pattern inequalities*)

The inverted pattern inequalities associated to a pattern $\mathcal{P}$ are the following, for any $\mathcal{X} \in \chi(\mathcal{P})$:

$$\sum_{x_{ij} \in \mathcal{X}} x_{ij} \geq card(\mathcal{P}) - rank(\mathcal{P}) \qquad (ipi(\mathcal{X}))$$

Consider a pattern $\mathcal{P}$, and a variable set $\mathcal{X} \in \chi(\mathcal{P})$. We show that $(pi(\mathcal{X}))$ for the ISCPKP modeled with $M_{R\text{-}ISCPK}$ and $(ipi(\mathcal{X}))$ for the ISCPKP modeled with M_{ISCPK} have the exact same dimension.

Property 22

Let $\overline{\mathcal{P}}$ be a pattern of the ISCPKP modeled with $M_{R\text{-}ISCPK}$. Let $\overline{\mathcal{X}} \in \chi(\overline{\mathcal{P}})$ and $\mathcal{X}$ be associated with $\overline{\mathcal{X}}$. For any solution of the ISCPKP modeled with $M_{R\text{-}ISCPK}$ verifying $(pi(\overline{\mathcal{X}}))$ to equality, there is a solution of the ISCPKP verifying $(ipi(\mathcal{X}))$ to equality.

Proof : Clearly, models $M_{R\text{-}ISCPK}$ and M_{ISCPK} are the same, as one can be obtained from the other through a variable substitution. Hence, for each feasible solution for one of these models, there is a feasible solution for the other one, by complementing and renumbering all variables.

■

As model $M_{R\text{-}ISCPK}$ has the same constraints as model M_{SCPK}, then the polyhedral results extend. In particular, one can transform any inequality defined for $M_{R\text{-}ISCPK}$ into an inequality for the M_{ISCPK} and guarantee the same dimension for the associated faces. Following these two steps, one can extend all results and algorithms designed for the SCPKP to the ISCPKP.

5.4 The discretized 1-HUC problem with resource windows as knapsack and covering inequalities

In this section, we aim to find a set of (I)SCPKPs that could share facet-defining inequalities with the 1-HUC^D problem. The idea is then to obtain facet-defining or strong inequalities for the 1-HUC^D problem from such (I)SCPKPs. For this purpose, we first define (I)SCPKP sub-problems of the 1-HUC^D problem, defined on a *time set* $\mathcal{T} \subseteq \{1,\dots,T\}$. In such a case, an instance of the (I)SCPKP ($\mathcal{T}$, J, W, V, C) can be defined with a number of groups $I = |\mathcal{T}|$. The idea is to formulate the (I)SCPKP sub-problems on the set of variables of the 1-HUC^D problem, and the knapsack or covering constraints apply only on variables for the groups corresponding to the time set $\mathcal{T}$.

Definition 33 (*SCPKP and ISCPKP as 1-HUC^D sub-problems*)

Consider a 1-HUC^D problem, and an (I)SCPKP. The (I)SCPKP is a 1-HUC^D sub-problem if its groups correspond to a time set $\mathcal{T}$. The items of the (I)SCPKP correspond to the operating points. The knapsack bound C corresponds to a knapsack (or covering) bound for the sum $\sum_{t\in\mathcal{T}} x_{ti}$ valid for the 1-HUC^D problem.

We show in the following how one can obtain the relevant values of C.

By definition, any (I)SCPKP is an 1-HUC^D sub-problem if all feasible solutions of the 1-HUC^D problem are also feasible for the (I)SCPKP. Hence, there is an exponential number of (I)SCPKPs that can be defined as 1-HUC^D sub-problems. Indeed, if an (I)SCPKP is an 1-HUC^D sub-problem, so is any similar (I)SCPKP with a higher knapsack constraint bound or a lower covering constraint bound. Hence, we define a dominance rule between (I)SCPKPs.

Definition 34 (*Dominance between SCPKPs and ISCPKPs*)

Consider two SCPKPs. One is dominated if its knapsack inequality is dominated by the knapsack inequality of the other SCPKP. Similarly, consider two ISCPKPs, one is dominated if its covering inequality is dominated by the covering inequality of the other ISCPKP.

We also define the characterizing (I)SCPKPs among a set of (I)SCPKPs, which are the not dominated and for which the knapsack/covering inequality is not trivially satisfied.

Definition 35 (*Characterizing SCPKPs and ISCPKPs*)

Let $\mathcal{K}$ be a set of (I)SCPKPs. Let ($\mathcal{T}$, C) be an SCPKP. This SCPKP is said to be characterizing among

$\mathcal{K}$ if it is not dominated by another SCPKP in $\mathcal{K}$ and if $C < |\mathcal{T}| \cdot \sum_{i=1}^{I} W_i$.

Let $(\mathcal{T}, C)$ be an ISCPKP. This ISCPKP is said to be characterizing among $\mathcal{K}$ if it is not dominated by another ISCPKP in $\mathcal{K}$ and if $C > 0$.

In the remainder of this chapter, we only consider (I)SCPKPs that are 1-HUC^D sub-problems. Hence, instances of the (I)SCPKPs are associated with the same 1-HUC^D, meaning that the only relevant difference between them lies in $\mathcal{T}$ and C. From now on, we denote an instance of the (I)SCPKP by $(\mathcal{T}, C)$.

5.4.1 Defining-(Inverted) Symmetric-weight Chain Precedence Knapsack Problem

Consider a time period t of the 1-HUC^D problem. It appears that the upper bound on the cumulated flow (3.2.7) for time period t, precedence constraints for time period $t' \in \{1,\dots,t\}$ (3.1.6) and the binary variables for time periods $t' \in \{1,\dots,t\}$ (3.1.14) correspond exactly to the constraints of an SCPKP. Similarly, the same set of constraints, with the lower bound on the cumulated flow (3.2.6) rather than the upper bound, corresponds exactly to the set on constraints of an ISCPKP. Such (I)SCPKPs are said to be the defining-(I)SCPKPs of the 1-HUC^D problem.

Definition 36 (*Defining-(I)SCPKPs of the 1-HUC^D problem*)

Let $(T, N, D, \beta^*, \alpha^*)$ be an instance of the 1-HUC^D problem. For each $1 \leq t \leq T$, SCPKP $(\mathcal{T} = \{1,2,\dots,t-1,t\}, C = \alpha_t^*)$ is said to be defining for the 1-HUC^D problem. Similarly, for each $1 \leq t \leq T$, ISCPKP $(\mathcal{T} = \{1,2,\dots,t-1,t\}, C = \beta_t^*)$ is said to be defining for the 1-HUC^D problem.

We define $\mathcal{K}_{def}$ the set of defining-(I)SCPKPs. Clearly, $\mathcal{K}_{def}$ contains exactly T defining-SCPKPs and T defining-ISCPKPs. Besides, one can represent all constraints of the 1-HUC^D problem with the (I)SCPKPs in $\mathcal{K}_{def}$. However, not all of them are relevant for the polyhedral study, meaning that some of them do not share facet-defining inequalities with the 1-HUC^D problem. We can identify the ones that are not relevant by establishing a dominance rule between two defining-(I)SCPKPs.

Property 23

Let $(\mathcal{T}, C)$ and $(\mathcal{T}', C')$ be two defining-SCPKPs. The former dominates the latter if $\mathcal{T} \supseteq \mathcal{T}'$ and $C \leq C'$.

Let $(\mathcal{T}, C)$ and $(\mathcal{T}', C')$ be two defining-ISCPKPs. The former dominates the latter if $\mathcal{T} \subseteq \mathcal{T}'$ and $C \geq C'$.

Proof : Let $(\mathcal{T}, C)$ and $(\mathcal{T}', C')$ be two defining-SCPKPs. Suppose $\mathcal{T} \supseteq \mathcal{T}'$ and $C \leq C'$. In such a case the former dominates the latter as it has a smaller upper bound for a larger set of variables.

Let $(\mathcal{T}, C)$ and $(\mathcal{T}', C')$ be two defining-ISCPKPs. Suppose $\mathcal{T} \subseteq \mathcal{T}'$ and $C \geq C'$. In such a case, the former dominates the latter as it has a larger lower bound for a smaller set of variables. ■

Example 23

Let an instance of the 1-HUC^D problem (3,3, [6, 5, 4], [0, 10, 10], [18, 20, 20]). ***Figure 5.1*** *represents the constraints of the 1-HUC^D problem. A square denoted by (t, i) corresponds to the operating point i at time period t. The order constraints are bottom up in this representation.*

As $T = 3$, there are 3 defining-SCPKPs, namely ({1}, 18), ({1, 2}, 20) and ({1, 2, 3}, 20). Similarly, there are 3 defining-ISCPKPs, namely ({1}, 0), ({1, 2}, 10) and ({1, 2, 3}, 10). Among them, we identify the ones that are also characterizing.

Defining-SCPKP ({1}, 18) cannot be characterizing, as $18 \geq (6 + 5 + 4) \cdot 1 = 15$.

Defining-SCPKP ({1, 2}, 20) cannot be characterizing, as defining-SCPKP ({1, 2, 3}, 20) satisfies $20 \geq 20$ *and* $\{1, 2\} \subseteq \{1, 2, 3\}$. *Hence, any solution of the 1-HUC^D satisfying the constraints of the latter also satisfies the constraints of the former.*

Defining-ISCPKP ({1}, 0) cannot be characterizing, as $0 \leq 0$.

Defining-ISCPKP ({1, 2, 3}, 10) cannot be characterizing, as defining-ISCPKP ({1, 2}, 10) satisfies $10 \leq 10$ *and* $\{1, 2, 3\} \supseteq \{1, 2\}$. *Hence, any solution of the 1-HUC^D satisfying the constraints of the latter also satisfies the constraints of the former.*

For this instance of the 1-HUC^D, among the defining-(I)SCPKPs only one defining-SCPKP and one defining-ISCPKP are characterizing. ***Figure 5.2*** *represents these characterizing defining-SCPKP and ISCPKP with similar representation as for* ***Figure 5.1***

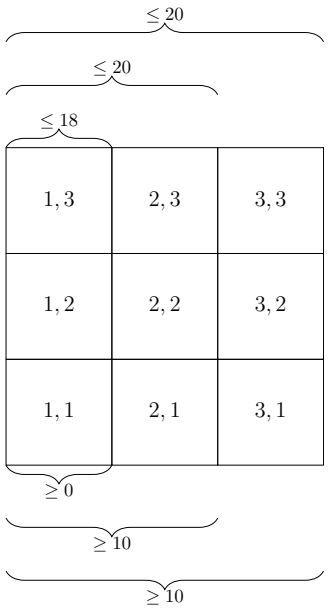


Figure 5.1: Representation of the 1-HUC^D problem as a collection of defining-SCPKPs and defining-ISCPKPs

Note that identifying the defining-SCPKPs and ISCPKPs that are characterizing among $\mathcal{K}_{def}$ can be achieved in a pre-processing phase. Indeed, only the bounds β^* and α^* of the 1-HUC^D problem

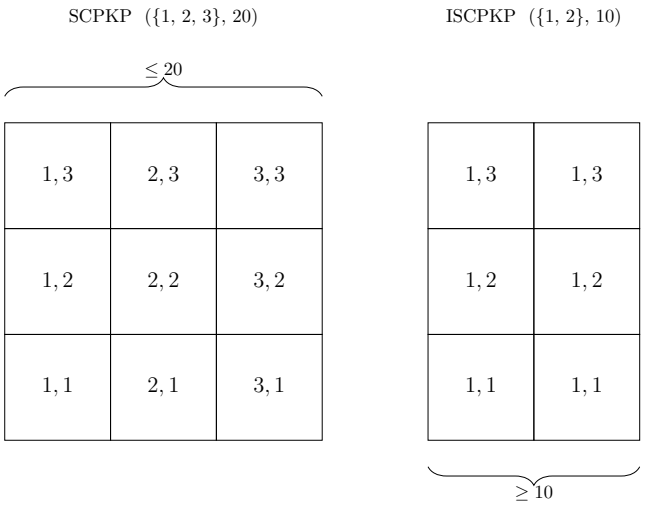


Figure 5.2: Representation of the characterizing defining-SCPKP and defining-ISCPKP

are required to compute them. Also, there are exactly T defining-SCPKPs and T defining-ISCPKPs, and obtaining the characterizing ones among $\mathcal{K}_{def}$ only requires to compare them pairwise, i.e., $\mathcal{O}(T^2)$ comparisons.

5.4.2 Induced (Inverted) Symmetric-weight Chain Precedence Knapsack Problem

As mentioned previously, from the defining-SCPKPs and ISCPKPs, it is possible to identify characterizing ones among $\mathcal{K}_{def}$. However, there are other (I)SCPKPs which can be relevant for the polyhedral study of the 1-HUC^D problem. In order to identify them, we define an Induced (I)SCPKP.

Definition 37 (*Induced SCPKP and ISCPKP*)

Consider an SCPKP $(\mathcal{T}, C)$ and an ISCPKP $(\mathcal{T}', C')$. The induced SCPKP is $(\mathcal{T} \setminus \mathcal{T}', C - (C' - |\mathcal{T}' \setminus \mathcal{T}| \cdot \sum_{i=1}^{N} D_i))$, and the induced ISCPKP is $(\mathcal{T}' \setminus \mathcal{T}, C' - C)$.

Note that, if $\mathcal{T} \subseteq \mathcal{T}'$ (resp. $\mathcal{T}' \subseteq \mathcal{T}$), then the induced SCPKP (resp. ISCPKP) is defined on an empty time set and can be ignored. We define $\mathcal{K}_{ind}$ the set of all induced (I)SCPKPs, obtained from defining-(I)SCPKPs or induced (I)SCPKPs.

Property 24

Consider an SCPKP and an ISCPKP, both sub-problems of the 1-HUC^D problem. An induced (I)SCPKP obtained from these two sub-problems is also a sub-problem of the 1-HUC^D problem.

Proof : Suppose the induced (I)SCPKP is not sub-problem of the 1-HUC^D problem. Then it means that there is a feasible solution of the 1-HUC^D problem which is infeasible for the (I)SCPKP. By construction of the induced (I)SCPKP, then the initial (I)SCPKP cannot be sub-problems of the 1-HUC^D problem. ■

For the following, if not specified otherwise, the characterizing (I)SCPKP are among $\mathcal{K}_{ind} \cup \mathcal{K}_{def}$.

The rationale behind the difference between the bound of the induced SCPKP and the one of the induced ISCPKP is that an upper-bound for a set of variables is valid for its subsets, whereas a lower bound for a set of variables is not. This phenomenon is illustrated in the following example.

Example 24

Consider an instance of the 1-HUC^D problem (3, 4, [2,1,3], β^*, α^*).

First, consider the defining-SCPKP ($\{1,2,3\}$, 15) and the ISCPKP ($\{1,2,4\}$, 20). In this case only the subset of time periods $\{1,2\}$ is in common. As 15 is an upper bound for the cumulative flow for the set of time periods $\{1,2,3\}$, it is also a valid upper bound for the cumulative flow for the set $\{1,2\}$. Hence, the cumulative flow cannot be larger than 15 for time periods $\{1,2\}$, but must be above 20 for time periods $\{1,2,4\}$. Consequently, the cumulative flow during time periods $\{4\}$ must be at least 20-15=5. We can derive the induced ISCPKP ($\{1,2,4\} \setminus \{1,2,3\}$, $20-15$)=($\{4\}$, 5).

Now consider the defining-SCPKP ($\{1,2,3\}$, 15) and ISCPKP ($\{1,2,4\}$, 10). Similarly, the subset $\{1,2\}$ is in common. In this case, 10 is a lower-bound for the cumulative flow for time periods $\{1,2,4\}$, but it is not a lower-bound for the cumulative flow for time periods $\{1,2\}$. Indeed, one can deliver $2+1+3$ at time period 4. This means that during time periods $\{1,2\}$, the cumulative flow is not necessarily above 10, but rather above $10-6=4$. Hence, the cumulative flow must be at least 4 during time periods $\{1,2\}$, and at most 15 during time periods $\{1,2,3\}$ Consequently, the cumulative flow cannot be larger than $15-4=11$ during time periods $\{3\}$. We can derive the induced SCPKP ($\{1,2,3\} \setminus \{1,2,4\}$, $15-(10-6)$)= ($\{3\}$, 11).

Before further studying the induced (I)SCPKPs, we first describe how the bounds of any 1-HUC^D sub-problem SCPKP and ISCPKP can be tightened. Tightening the bounds is necessary because if the bounds of an SCPKP and ISCPKP are not tight, the induced (I)SCPKP will cumulate the slack, which can make the induced (I)SCPKP irrelevant from a polyhedral study point of view. We define more clearly what we consider to be an under-constrained (I)SCPKP.

Definition 38 (*Under-constrained SCPKP and ISCPKP*)

Consider two SCPKPs sub-problems, ($\mathcal{T}$, C) and ($\mathcal{T}'$, C') respectively. The SCPKP ($\mathcal{T}$, C) is under-constrained if its set of feasible solution is a super set of the one of SCPKP ($\mathcal{T}$, C'). The definition is similar for an ISCPKP.

In the case of an under-constrained (I)SCPKP, the polyhedral results may become irrelevant for the 1-HUC^D problem. The following example illustrates such a case.

Example 25

Consider an instance of the 1-HUC^D problem (4, 3, [6, 3, 2], [18, 18, 18, 18], [0, 1, 1, 1]. The only characterizing defining-SCPKP is ({1, 2, 3, 4}, 18) and the only characterizing defining-ISCPKP is ({1, 2}, 1). An induced SCPKP would be ({3, 4}, 17), which we show in the following to be under-constrained. Due to the flows $D = [6,3,2]$, in order to verify the bound $\beta^*_2 = 1$, it is necessary to

deliver at least 6 units. As such it is valid to consider the ISCPKP ({1, 2}, 6). As it has the same feasible solution as the ISCPKP ({1, 2}, 1), the facet-defining inequalities are the same. However, the induced SCPKP would be ({3, 4}, 12) instead of ({3, 4}, 17), and they do not allow the same set of feasible solutions. Delivering 6 at time period 3, then 11 at time period 4 is valid for the latter but not for the former. Hence, the polyhedral results of SCPKP ({3, 4}, 17) yield valid inequalities, but they are clearly dominated by the ones from SCPKP ({3, 4}, 12).

Providing the tightest bound of a defining-SCPKP for time set $\mathcal{T}$ can be done by finding the solution of the 1-HUC^D problem, maximizing the sum of the flows for all time period $t \in \mathcal{T}$. In such a case, the value of every operating points $i \leq N$ are $\Psi_{t,i} = D_t$ for time periods $t \in \mathcal{T}$ and $\Psi_{t,i} = 0$ otherwise. Hence, one needs to solve the 1-HUC^D problem as follows:

$$\max \sum_{t \in \mathcal{T}} \sum_{j=1}^{N} D_i \cdot x_{ij}$$

$$\sum_{t=1}^{t'} \sum_{i=1}^{N} D_i x_{t,i} \geq \beta_{t'} \quad \forall t' \leq T \tag{3.2.6}$$

$$\sum_{t=1}^{t'} \sum_{i=1}^{N} D_i x_{t,i} \leq \alpha_{t'} \quad \forall t' \leq T \tag{3.2.7}$$

$$x_{t,i} \geq x_{t,i+1}, \qquad \forall t \leq T, \forall i \in \{0, \ldots, N-1\} \tag{3.1.6}$$

$$x_{t,i} \in \{0,1\}, \qquad \forall t \leq T, \forall i \in \mathcal{N} \tag{3.1.14}$$

Tightening the bound of an ISCPKP would be similar, but minimizing the objective function.

Note that this problem is a 1-HUC^D problem, but for which the values are identical to the flows. As the flows are symmetric, so are the values. However, we remain in a case where the feasible solutions are not symmetric due to bounds β^* and α^* which are non-constant in general. Preliminary results show that this particular case of the 1-HUC^D problem are solved more quickly than the generic 1-HUC^D problem by MILP solvers. Moreover, we show later in this section that tightening the bounds is not required on all (I)SCPKP, but only on the defining ones.

Once the bounds of the SCPKP and ISCPKP are tightened, one can derive the polyhedral results to help solving the 1-HUC^D problem. Clearly, the dominance property defined in **Property 23** directly extends to any (I)SCPKP. Due to the existence of an induced (I)SCPKP for any time set $\mathcal{T}$, it is also possible for an SCPKP (resp. ISCPKP) to be dominated by a sum of several SCPKP (resp. ISCPKP).

Property 25

Consider K different SCPKPs ($\mathcal{T}_k$, C_k) for each $k \leq K$, such that $\mathcal{T}_k \cap \mathcal{T}_{k'} = \emptyset$ for any $k \neq k'$. Consider the SCPKP ($\mathcal{T}$, C) with $\mathcal{T} = \bigcup_{k=1}^{K} \mathcal{T}_k$ and $C = \sum_{k=1}^{K} C_k$. Then SCPKP ($\mathcal{T}'$, C') is dominated by SCPKP ($\mathcal{T}$, C) if $\mathcal{T} \supseteq \mathcal{T}'$ and $C \leq C'$.

Consider K different ISCPKPs ($\mathcal{T}_k$, C_k) for each $k \leq K$, such that $\mathcal{T}_k \cap \mathcal{T}_{k'} = \emptyset$ for any $k \neq k'$. Consider the ISCPKP ($\mathcal{T}$, C) with $\mathcal{T} = \bigcup_{k=1}^{K} \mathcal{T}_k$ and $C = \sum_{k=1}^{K} C_k$. Then ISCPKP ($\mathcal{T}'$, C') is dominated by ISCPKP

$(\mathcal{T}, C)$ if $\mathcal{T} \subseteq \mathcal{T}'$ and $C \geq C'$.

Proof : Consider K different SCPKPs: $(\mathcal{T}_k, C_k)$ for each $k \leq K$. Suppose that $\mathcal{T}_k \cap \mathcal{T}_{k'} = \emptyset$ for any $k \neq k'$. Consider the SCPKP $(\mathcal{T}, C)$ with $\mathcal{T} = \bigcup_{k=1}^{K} \mathcal{T}_k$ and $C = \sum_{k=1}^{K} C_k$. As $\mathcal{T}_k \cap \mathcal{T}_{k'} = \emptyset$ for any $k \neq k'$, then this new SCPKP is valid. Using **Property 23** SCPKP $(\mathcal{T}, C)$ dominates any SCPKP $(\mathcal{T}', C')$ with $\mathcal{T} \supseteq \mathcal{T}'$ and $C \leq C'$.

A similar proof can be done for the dominance of the ISCPKP. ■

We show in the following example the dominance between one (I)SCPKP and multiple (I)SCPKPs.

Example 26

Let an instance of the 1-HUC^D problem be (3, 3, [2, 2, 3], [5, 5, 14], [11, 11, 15]).

From defining-SCPKP ({1, 2}, 11) and defining-ISCPKP ({1}, 5) the induced SCPKP ({2}, 6) is characterizing.

From defining-SCPKP ({1, 2}, 11) and defining-ISCPKP ({1, 2, 3}, 14) the induced ISCPKP ({3}, 3) is characterizing.

From induced SCPKP ({2}, 6) and the defining-ISCPKP ({1, 2, 3}, 14) the induced ISCPKP ({1,3}, 8) is not characterizing. Indeed, there are already the defining-ISCPKP ({1}, 5) and the induced ISCPKP ({3}, 3). Hence, as $\{1,3\} \supseteq \{1\} \cup \{3\}$ and $8 \geq 5 + 3$, then ISCPKP ({1,3}, 8) is not characterizing.

As SCPKP and ISCPKP are defined for a time set $\mathcal{T}$, there can be an exponential number of characterizing induced SCPKP and ISCPKP for a 1-HUC^D problem. In such a case, it would mean to take into account the inequalities of an exponential number of SCPKP and ISCPKP, which is unrealistic. Fortunately, we can prove that only SCPKP and ISCPKP defined on a continuous time set can be characterizing.

Property 26 (*Characterizing SCPKP and ISCPKP have continuous time sets*)

An (I)SCPKP $(\mathcal{T}, C)$ is characterizing among $\mathcal{K}_{ind} \cup \mathcal{K}_{def}$ only if $\mathcal{T}$ is continuous, i.e., there exists t_1 and t_2 such that $\mathcal{T} = \{t_1, t_1 + 1, \ldots, t_2 - 1, t_2\}$.

Proof : Consider an instance of the 1-HUC^D problem. Consider an induced SCPKP $(\mathcal{T}, C)$ and suppose it to be characterizing. Suppose that time set $\mathcal{T} = \{t_1 + 1, \ldots, t_2\} \cup \{t_3 + 1, \ldots, t_4\}$ with $1 \leq t_1 < t_2 < t_3 < t_4 \leq T$.

By definition, there are two defining-SCPKPs $(\{1, \ldots, t_2\}, \alpha^*_{t_2})$ and $(\{1, \ldots, t_4\}, \alpha^*_{t_4})$ as well as two defining-ISCPKPs $(\{1, \ldots, t_1\}, \beta^*_{t_1})$ and $(\{1, \ldots, t_3\}, \beta^*_{t_3})$. From defining-SCPKP $(\{1, \ldots, t_4\}, \alpha^*_{t_4})$ and defining-ISCPKP $(\{1, \ldots, t_1\}, \beta^*_{t_1})$, one can induce the SCPKP $(\{t_1 + 1, \ldots, t_4\}, \alpha^*_{t_4} - \beta^*_{t_1})$. Similarly, from defining-ISCPKP $(\{1, \ldots, t_3\}, \beta^*_{t_3})$ and defining-SCPKP $(\{1, \ldots, t_2\}, \alpha^*_{t_2})$, one can induce the ISCPKP $(\{t_2 + 1, \ldots, t_3\}, \beta^*_{t_3} - \alpha^*_{t_2})$.

From induced SCPKP $(t_1 + 1, \ldots, t_4, \alpha^*_{t_4} - \beta^*_{t_1})$ and ISCPKP $(\{t_2 + 1, \ldots, t_3\}, \beta^*_{t_3} - \alpha^*_{t_2})$, one can induce SCPKP $(\mathcal{T}, C)$. Hence, we deduce $C = \alpha^*_{t_4} - \beta^*_{t_3} + \alpha^*_{t_2} - \beta^*_{t_1}$

Besides, with the defining-SCPKP $(\{1, \ldots, t_4\}, \alpha^*_{t_4})$ and defining-ISCPKP $(\{1, \ldots, t_3\}, \beta^*_{t_3})$ one can induce SCPKP $(\{t_3 + 1, \ldots, t_4\}, \alpha^*_{t_4} - \beta^*_{t_3})$. Similarly, with the defining-SCPKP $(\{1, \ldots, t_2\}, \alpha^*_{t_2})$ and defining-ISCPKP $(\{1, \ldots, t_1, \alpha^*_{t_1})$ one can induce SCPKP $(\{t_1 + 1, \ldots, t_2\}, \alpha^*_{t_2} - \beta^*_{t_1})$.

These two induced SCPKP dominate the SCPKP $(\mathcal{T}, C)$ where $C = \alpha^*_{t_4} - \beta^*_{t_3} + \alpha^*_{t_2} - \beta^*_{t_1}$ and $\mathcal{T} = \{t_1 + 1, \ldots, t_2\} \cup \{t_3 + 1, \ldots, t_4\}$. Hence, SCPKP $(\mathcal{T}, C)$ cannot be characterizing.

A similar proof can be done for an induced ISCPKP. ■

With the characterizing SCPKP and ISCPKP restricted to a continuous time set, we can provide a polynomial upper bound on their number.

Property 27 (*Polynomial number of characterizing SCPKPs and ISCPKPs*)

There are at most $\mathcal{O}(T^2)$ characterizing SCPKPs (resp. ISCPKPs) among $\mathcal{K}_{ind} \cup \mathcal{K}_{def}$.

Proof : An SCPKP (resp. ISCPKP) is characterizing only for a continuous time set. Hence, for any pair of time periods (t_1, t_2) with $t_1 \leq t_2$ there is exactly one SCPKP (resp. ISCPKP) being $(\{t_1, \dots, t_2\}, C)$. The number of such pairs being T^2, there are at most T^2 characterizing SCPKPs (resp. ISCPKPs). ■

As there can be up to T^2 characterizing SCPKPs and T^2 characterizing ISCPKPs, one can obtain all of them by naively comparing each characterizing SCPKP and ISCPKP. This leads to a total of $T^2 \cdot T^2 = T^4$ comparisons, which could become difficult to compute for large instances. In the following, we prove that any characterizing ISCPKP and SCPKP can be induced from the defining-ISCPKPs and defining-SCPKPs. As there are exactly T such SCPKPs and ISCPKPs, only $T \cdot T = T^2$ comparisons are necessary.

To do such a proof, we first introduce conditions to induce a characterizing (I)SCPKP.

Lemma 2 (*SCPKP inducing conditions*)

Consider an SCPKP $(\mathcal{T}, C)$ with $\mathcal{T} = \{t_1, \dots, t_2\}$ and an ISCPKP $(\mathcal{T}', C')$ with $\mathcal{T}' = \{t'_1, \dots, t'_2\}$. The induced SCPKP is characterizing among $\mathcal{K}_{ind} \cup \mathcal{K}_{def}$ only if either $t'_1 \leq t_1 \leq t'_2 < t_2$ or $t_1 < t'_1 \leq t_2 \leq t'_2$.

Proof : In order to obtain an induced SCPKP from SCPKP $(\mathcal{T}, C)$ and ISCPKP $(\mathcal{T}', C')$, some conditions on their time sets must be verified.

First, $\mathcal{T} \cap \mathcal{T}' \neq \emptyset$ otherwise the induced SCPKP would be $(\mathcal{T}, C - (C' - (|\mathcal{T}'| \cdot \sum_{i=1}^{J} D_i)$. Clearly, $C' \leq |\mathcal{T}'| \cdot \sum_{i=1}^{J} D_i$, otherwise ISCPKP $(\mathcal{T}', C')$ would not have any feasible solution. Hence $C - (C' - (|\mathcal{T}') \sum_{i=1}^{J} D_i \geq C$, and the induced SCPKP would be dominated by SCPKP $(\mathcal{T}, C)$. This means that $t_1 \leq t'_2$ and $t'_1 \leq t_2$.

Second, recall that an induced SCPKP is defined for time set $\mathcal{T} \setminus \mathcal{T}'$. This means that if $\mathcal{T} \subseteq \mathcal{T}'$ the induced SCPKP is defined on an empty time set. Thus, either $t_1 < t'_1$ or $t'_2 < t_2$.

Third, if $t_1 < t'_1$ and $t'_2 < t_2$, the induced SCPKP is defined on the time set $\{t_1, \dots, t'_1 - 1\} \cup \{t'_2 + 1, \dots, t_2\}$. In such a case, the induced SCPKP cannot be characterizing as proven in **Property 26**. Consequently, if $t_1 < t'_1$, then $t_2 \leq t'_2$, and if $t'_2 < t_2$, then $t'_1 \leq t_1$.

The two possibilities satisfying these three conditions are $t'_1 \leq t_1 \leq t'_2 < t_2$ and $t_1 < t'_1 \leq t_2 \leq t'_2$. ■

Lemma 3 (*ISCPKP inducing conditions*)

Consider an SCPKP $(\mathcal{T}, C)$ with $\mathcal{T} = \{t_1, \dots, t_2\}$ and an ISCPKP $(\mathcal{T}', C')$ with $\mathcal{T}' = \{t'_1, \dots, t'_2\}$. The induced ISCPKP is characterizing among $\mathcal{K}_{ind} \cup \mathcal{K}_{def}$ only if either $t'_1 < t_1 \leq t'_2 \leq t_2$ or $t_1 \leq t'_1 \leq t_2 < t'_2$.

We do not detail the proof as it is similar as the one for **Lemma 2**.

Property 28

An induced ISCPKP is characterizing among $\mathcal{K}_{ind} \cup \mathcal{K}_{def}$ only if it is induced from a defining-SCPKP and a defining-ISCPKP.

Proof : This proof is illustrated by **Figure 5.3**. In this figure, a square denoted by t represents time period t. Curly brackets above the squares represent the time sets of an SCPKP as well as its knapsack bound. Similarly, curly brackets below the squares represent the time sets of an ISCPKP as well as its covering bound.

Consider an instance of the 1-HUC^D problem. Consider an induced ISCPKP ($\mathcal{T}$, C), induced from a defining-SCPKP ($\{1,\dots,t_2\}$, $\alpha^*_{t_2}$) and an induced ISCPKP ($\mathcal{T}'$, C') with $\mathcal{T}' = \{t'_1,\dots,t'_2\}$. Suppose that ISCPKP ($\mathcal{T}$, C) is characterizing. As proven in **Lemma 3** in order for ISCPKP ($\mathcal{T}$, C) to be characterizing, then $1 \leq t'_1 \leq t_2 < t'_2$ or $t'_1 < 1 \leq t'_2 \leq t_2$. Clearly, it is impossible to have $t'_1 < 1$, hence we consider the former case. Besides, as ISCPKP ($\mathcal{T}'$, C') is induced, hence $t'_1 > 1$, otherwise it would be a defining-ISCPKP. We deduce $\mathcal{T} = \{t_2+1,\dots,t'_2\}$ and $C = C' - \alpha^*_{t_2}$ as illustrated in **Figure 5.3a**.

By construction, there is a defining-SCPKP ($\{1,\dots,t'_1-1\}$, $\alpha^*_{t'_1-1}$) and a defining-ISCPKP ($\{1,\dots,t'_2\}$, $\beta^*_{t'_2}$). Consequently, we deduce $C' = \beta^*_{t'_2} - \alpha^*_{t'_1-1}$ as depicted in **Figure 5.3b**.

By construction, there is a defining-SCPKP ($\{1,\dots,t_2\}$, $\alpha^*_{t_2}$) and a defining-ISCPKP ($\{1,\dots,t'_2\}$, $\beta^*_{t'_2}$). From them, we can induce ISCPKP ($\mathcal{T}$, $\beta^*_{t'_2} - \alpha^*_{t_2}$) as shown in **Figure 5.3c**.

As we suppose ISCPKP ($\mathcal{T}$, C) to be characterizing, then $C > \beta^*_{t'_2} - \alpha^*_{t_2}$:

$$\begin{aligned} C &> \beta^*_{t'_2} - \alpha^*_{t_2} \\ \beta^*_{t'_2} - \alpha^*_{t'_1-1} - \alpha^*_{t_2} &> \beta^*_{t'_2} - \alpha^*_{t_2} \\ \alpha^*_{t'_1-1} &< 0 \end{aligned}$$

As such, the instance of the 1-HUC^D is infeasible as there is a negative upper bound. Hence, ISCPKP ($\mathcal{T}$, C) cannot be characterizing.

A similar proof can be made for an ISCPKP induced from a defining-ISCPKP and an induced SCPKP, or from and induced SCPKP and ISCPKP. ■

Property 29

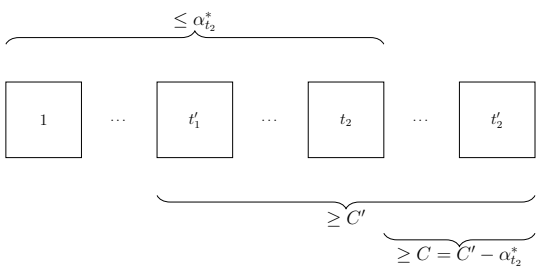
An induced SCPKP is characterizing among $\mathcal{K}_{ind} \cup \mathcal{K}_{def}$ only if it is induced from a defining-SCPKP and a defining-ISCPKP.

We do not detail the proof as it is very similar to the one for **Property 28**.

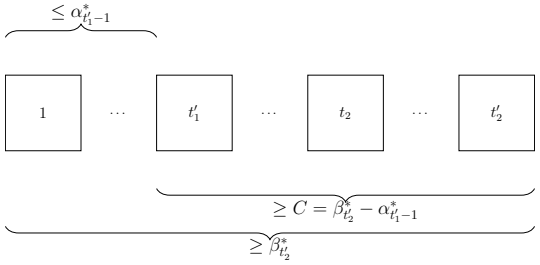
Now that we proved that all characterizing (I)SCPKP are either defining, or induced from the defining ones, we show that bound tightening is needed only on defining-(I)SCPKP.

Property 30

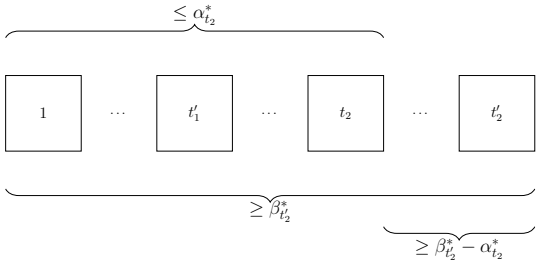
Consider a defining-SCPKP and a defining-ISCPKP. The induced (I)SCPKP from these defining-(I)SCPKP cannot be under-constrained if the bounds of the defining-(I)SCPKP are tight.



(a) Deducing C with respect to $\alpha^*_{t_2}$ and C'



(b) Deducing C' with respect to $\beta^*_{t'_2}$ and $\alpha^*_{t'_1-1}$



(c) Inducing another SCPKP for time set $\mathcal{T}$

Figure 5.3: Illustrations for **Property 28**

Proof : Clearly, tightening the bound of a defining-SCPKP or a defining-ISCPKP does not change its feasible solutions, hence facet-defining inequalities are the same with and without a tight bound. However, as shown in Example 25, tightening the bound of defining-SCPKPs and defining-ISCPKPs can change the feasible solutions of the induced ones. Also, as proven it **Property 28** and **29**, characterizing SCPKP and ISCPKP can only be induced from defining ones. Hence, if the bounds of the defining-SCPKP and ISCPKP are tightened, then none of the characterizing (I)SCPKP is under-constrained. If there is an under-constrained characterizing (I)SCPKP, then it leads to a contradiction, as one of the defining-(I)SCPKP cannot be tightened. ■

Finally, we can also show that all characterizing defining-(I)SCPKP among $\mathcal{K}_{def}$ are also characterizing among $\mathcal{K}_{def} \cup \mathcal{K}_{ind}$.

Property 31

Consider a defining-SCPKP. If it is characterizing among $\mathcal{K}_{def}$, it is also characterizing among $\mathcal{K}_{def} \cup \mathcal{K}_{ind}$.

Proof : Consider a defining-SCPKP $(\mathcal{T}, C)$, characterizing among $\mathcal{K}_{def}$. Suppose there is an induced SCPKP $(\mathcal{T}', C')$ which dominates defining-SCPKP $(\mathcal{T}, C)$. This means that $\mathcal{T}' \supseteq \mathcal{T}$ and $C' \leq C$. The induced SCPKP is from a defining-SCPKP and a defining-ISCPKP, respectively on continuous time sets $\mathcal{T}_1 = \{1, \ldots, t_1\}$ and $\mathcal{T}_2 = \{1, \ldots, t_2\}$. Hence, $1 \notin \mathcal{T}'$ otherwise there is a contradiction. Also, $1 \in \mathcal{T}$ as SCPKP $(\mathcal{T}, C)$ is defining, meaning that $\mathcal{T}' \supseteq \mathcal{T}$ yields a contradiction. Hence, an induced SCPKP cannot dominate a defining-SCPKP.

Consider a defining-ISCPKP $(\mathcal{T}, C)$, characterizing among $\mathcal{K}_{def}$. Suppose there is an induced ISCPKP $(\mathcal{T}', C')$ which dominates defining-ISCPKP $(\mathcal{T}, C)$. This means that $\mathcal{T}' \subseteq \mathcal{T}$ and $C' \geq C$. In order for $\mathcal{T}' \subseteq \mathcal{T}$, there must be a defining-ISCPKP $(\mathcal{T}'', C'')$ with $\mathcal{T}'' \subseteq \mathcal{T}$. By definition of an induced ISCPKP, $C' \leq C''$.

Consequently, there is a contradiction as defining-ISCPKP ($\mathcal{T}$, C) cannot be characterizing among $\mathcal{K}_{def}$, as it is dominated by defining-ISCPKP ($\mathcal{T}''$, C'').

As all induced SCPKP are from defining ones, a combination of defining-SCPKP and induced SCPKP cannot dominate the defining-SCPKP, and similarly for the ISCPKP. Consequently, the defining-SCPKP that are characterizing for $\mathcal{K}_{def}$ are also characterizing for $\mathcal{K}_{def} \cup \mathcal{K}_{ind}$. ■

In this section, we defined sets of (I)SCPKPs which are sub-problems of the 1-HUC^D problem : defining-(I)SCPKPs and induced (I)SCPKPs. From these sets, we can easily find the ones which are characterizing, as it only requires to compare the ones on a continuous time set as proven in **Property 26**. As such, obtaining the characterizing (I)SCPKPs only necessitates a quadratic number of comparisons. Based on the set of characterizing (I)SCPKPs, we can generate valid inequalities for each of these sub-problems using the study presented in **Chapter 4**, which gives also strong inequalities for the 1-HUC^D problem.

In the next section, we generalize polyhedral symmetries to the 1-HUC^D problem in order to identify which permutations of these inequalities are relevant. On this basis, we also generalize the concept of patterns to the 1-HUC^D problem.

5.5 Polyhedral results for the discretized 1-Hydro Unit Commitment problem

In this section, we address the polyhedral study of the 1-HUC^D problem. Recall that the constraints of the 1-HUC^D problem can be defined as an intersection of several (I)SCPKPs. Hence, we aim to extend the polyhedral results of the SCPKP and the ISCPKP to the 1-HUC^D problem. As polyhedral results of the (I)SCPKP involve patterns defined to handle polyhedral symmetries, we first study the symmetries of the 1-HUC^D problem.

5.5.1 Polyhedral symmetries for the discretized 1-Hydro Unit Commitment problem

Due to the presence of several upper and lower bounds, the polyhedral symmetries of the SCPKP do not entirely hold for the 1-HUC^D problem. However, we can identify several sets of permutations for which the polyhedral symmetries extend to the 1-HUC^D problem.

Definition 39 (*Symmetric time set for the 1-HUC^D problem*)

A time set $\mathcal{T}$ is symmetric for the 1-HUC^D problem if polyhedral symmetry holds for any permutation of time periods in $\mathcal{T}$.

We show that polyhedral symmetries hold for sets of time periods where bounds β_t^* and α_t^* are constants.

Theorem 12 (*Polyhedral symmetries of the 1-HUC^D problem*)

Let $\mathcal{T}$ be a time set. If for all characterizing defining-(I)SCPKPs on a time set $\mathcal{T}'$, time set $\mathcal{T}$ is such

that $\mathcal{T} \subseteq \mathcal{T}'$ or $\mathcal{T} \cap \mathcal{T}' = \emptyset$, then $\mathcal{T}$ is a symmetric time set for the 1-HUC^D problem.

Proof : Let $\mathcal{T}$ be a time set. Suppose that for each characterizing defining-(I)SCPKP on a time set $\mathcal{T}'$, then $\mathcal{T} \subseteq \mathcal{T}'$ or $\mathcal{T} \cap \mathcal{T}' = \emptyset$. Hence, bounds β_t^* and α_t^* are constants for time periods $t \in \mathcal{T}$. Consequently, for any feasible solution of the 1-HUC^D problem, a permutation of the time periods in $\mathcal{T}$ yield to another feasible solution of the 1-HUC^D problem. ■

Theorem 12 is particularly important, as for any facet-defining inequality, we can permute the time indices of a symmetric time set and obtain another facet-defining inequality.

Corollary 1 (*Symmetry of facet-defining inequalities for the 1-HUC^D problem*)

Let $\mathcal{T}$ be a symmetric time set. Then, for any facet-defining inequality, any permutation of $\mathcal{T}$ yield another facet-defining inequality.

Proof : The proof is similar to the one in **Property 5**. Let $\mathcal{T}$ be a time set for which polyhedral symmetries hold for the 1-HUC^D problem. Consider a facet-defining inequality for the 1-HUC^D problem. Then, there are n affinely dependent points, for which the solutions satisfy such an inequality to equality. Hence, one can prove any symmetry of a facet-defining inequality to also be facet-defining, as it suffices to deduce the n valid solutions following the same permutation of $\mathcal{T}$. These new n points are necessarily affinely independent as they all undergo the exact same permutation of the time periods. ■

Note that there are some special cases of the 1-HUC^D problem for which there are no polyhedral symmetries. This is when, for every $t \leq T$, either $\beta_t^* < \beta_{t+1}^*$ or $\alpha_t^* < \alpha_{t+1}^*$. In such a case, if an inequality is facet-defining, it is possible that none of its symmetry is also facet-defining.

For any other case, there are symmetric time sets, meaning that the polyhedral symmetries extend to the 1-HUC^D problem. Using similar structures to the patterns defined in **Section 4.4** is then relevant to handle the polyhedral symmetries. Moreover, we show in the following that there are large sets of inequalities, that are valid for any permutation. Such patterns can also be used to handle these sets of valid inequalities, as they can still be used to cut fractional points in a Branch & Cut scheme.

Theorem 12 provides valid sets of time periods for which polyhedral symmetries occur. In practice however, these sets defined in **Theorem 12** do not capture all polyhedral symmetries, as shown in the following **Example 27**.

Example 27

*Let (4, 3, [6,3,2], β^*, α^*) be an instance of the 1-HUC^D problem, with characterizing defining-SCPKP ({1, 2, 3, 4}, 18) and defining-ISCPKP ({1, 2}, 1), which respectively become ({1, 2, 3, 4}, 18) and ({1, 2}, 6) once their bounds are tightened. The only induced characterizing SCPKP is ({3, 4}, 12), which has the same bound once tightened. In this case, polyhedral symmetries as defined in **Theorem 12** exist for the sets of time periods $\{1, 2\}$ and $\{3, 4\}$. In the following, we exhibit facet-defining inequalities for which the polyhedral symmetry apply.*

For the set of time periods $\{1,2\}$*, polyhedral symmetries appear for inequalities:*

$$x_{1,2} + x_{2,3} + x_{3,3} + x_{4,3} \leq 1$$
$$x_{1,3} + x_{2,2} + x_{3,3} + x_{4,3} \leq 1$$

$$x_{1,1} + x_{1,3} + x_{2,2} + x_{3,2} + x_{4,2} \leq 2$$
$$x_{1,2} + x_{2,1} + x_{2,3} + x_{3,2} + x_{4,2} \leq 2$$

$$x_{1,1} + x_{1,2} + x_{2,1} + x_{2,3} + x_{3,1} + x_{3,3} + x_{4,1} + x_{4,3} \leq 3$$
$$x_{1,1} + x_{1,3} + x_{2,1} + x_{2,2} + x_{3,1} + x_{3,3} + x_{4,1} + x_{4,3} \leq 3$$

For the set of time periods $\{3,4\}$*, polyhedral symmetries appear for inequalities:*

$$x_{3,1} + x_{4,2} \leq 1$$
$$x_{3,2} + x_{4,1} \leq 1$$

However, there are inequalities for which permutations go beyond the time sets defined in ***Theorem 12****, which are* $\{1,2\}$ *and* $\{3,4\}$ *in this example:*

$$x_{1,1} + x_{2,1} + x_{3,2} + x_{4,2} \leq 2$$
$$x_{1,1} + x_{2,2} + x_{3,1} + x_{4,2} \leq 2$$
$$x_{1,1} + x_{2,2} + x_{3,2} + x_{4,1} \leq 2$$
$$x_{1,2} + x_{2,1} + x_{3,1} + x_{4,2} \leq 2$$
$$x_{1,2} + x_{2,1} + x_{3,2} + x_{4,1} \leq 2$$

In this case, all permutations are allowed but $x_{1,1} + x_{2,1} + x_{3,2} + x_{4,2} \leq 2$*.*

The sets identified in **Theorem 12** remain sufficient to motivate generalizing the patterns to the 1-HUC^D problem. In this thesis, we did not generalize the polyhedral symmetries in **Theorem 12** to capture all permutations, such as the last case shown in **Example 27**.

5.5.2 Generalized patterns

Given that polyhedral symmetries hold also to some extent in the 1-HUC^D problem, the patterns can be generalized quite directly to this problem. The only difference is that the patterns do not consider all time periods, but are defined for a subset of time periods, whereas for the (I)SCPKP, patterns where defined for all groups.

Definition 40 (*Generalized pattern*)

Let $\mathcal{T}$ be a time set. A generalized pattern $\mathcal{P}$ is a collection of $|\mathcal{T}|$ sets $S_t(\mathcal{P}) \subseteq \{1,\dots,N\}$, $t \in \mathcal{T}$, defined for the time set $\mathcal{T}$.

As a generalized pattern is defined on a time set $\mathcal{T}$, we must also generalize the variable set of a pattern as well as the rank.

Definition 41 (*Variable set $\mathcal{X}$ associated with $\mathcal{P}$*)

A variable set $\mathcal{X}$ is associated with generalized pattern $\mathcal{P}$ and a permutation π of $\mathcal{T}$ if:
$x_{ij} \in \mathcal{X} \Leftrightarrow j \in S_{\pi(i)}(\mathcal{P})$.

Definition 42 (*$rank(\mathcal{P})$*)

The rank of a generalized pattern $\mathcal{P}$ is

$$rank(\mathcal{P}) = \max_{\mathcal{X} \in \mathcal{X}(\mathcal{P})} \left\{ \max \sum_{x_{ij} \in \mathcal{X}} x_{ij} : \text{satisfying } (3.2.6), (3.2.7), (3.1.6), (3.1.14) \right\}.$$

Two concepts can be extended directly to the generalized patterns, namely the cardinality and the (inverted) pattern inequalities.

Remark 6

For the (I)SCPKP there was no ambiguity as all patterns were either only associated with pattern inequalities for the SCPKP, or only associated with inverted pattern inequalities for the ISCPKP. However, for the 1-HUC^D problem, there can be patterns associated with both. To avoid confusion, we also introduce Generalized Inverted Patterns. As such, we can make the distinction between a Generalized Pattern (GP) and a Generalized Inverted Patterns (GIP), respectively with pattern inequalities and inverted pattern inequalities.

The aim is now to study the relationship between pattern inequalities for the (I)SCPKP and GP inequalities for the 1-HUC^D problem. For this purpose, we look at the following example, providing inequalities from the convex hull of a small instance of the 1-HUC^D problem.

Example 28

*Consider the same instance as in the previous **Example 27**. From the convex hull of this instance, we observe three sets of 0-1 facet-defining inequalities.*

The first set of inequalities contains (inverted) pattern inequalities associated with patterns-facets coming from characterizing (I)SCPKP for which all permutations yield another facet-defining inequality of the 1-HUC^D problem. In this example we obtain the following inequalities:

- *Pattern {{1}, {1}} for ISCPKP ({1, 2}, 6)*

$$x_{1,1} + x_{2,1} \geq 1$$

- *Pattern {{1}, {2}} for SCPKP ({3, 4}, 12)*

$$x_{3,1} + x_{4,2} \leq 1$$
$$x_{3,2} + x_{4,1} \leq 1$$

- *Pattern {{2, 3}, {2, 3}, {2, 3}, {2, 3}} for SCPKP ({1, 2, 3, 4}, 18)*

$$x_{1,2} + x_{1,3} + x_{2,2} + x_{2,3} + x_{3,2} + x_{3,3} + x_{4,2} + x_{4,3} \leq 2$$

The second set of inequalities contains (inverted) pattern inequalities associated with patterns-facet coming from characterizing (I)SCPKP, for which only some permutations yield another facet-defining inequality of the 1-HUC^D problem. In this case, we obtain the following:

- *Pattern {{2}, {3}, {3}, {3}} for ISCPKP ({1, 2, 3, 4}, 18)*

$$x_{1,2} + x_{2,3} + x_{3,3} + x_{4,3} \leq 1$$
$$x_{1,3} + x_{2,2} + x_{3,3} + x_{4,3} \leq 1$$

- *Pattern {{1}, {1}, {2}, {2}} related to SCPKP ({1, 2, 3, 4}, 18)*

$$x_{1,1} + x_{2,1} + x_{3,2} + x_{4,2} \leq 2$$
$$x_{1,1} + x_{2,2} + x_{3,1} + x_{4,2} \leq 2$$
$$x_{1,1} + x_{2,2} + x_{3,2} + x_{4,1} \leq 2$$
$$x_{1,2} + x_{2,1} + x_{3,1} + x_{4,2} \leq 2$$
$$x_{1,2} + x_{2,1} + x_{3,2} + x_{4,1} \leq 2$$

- *Pattern {{1, 3}, {2}, {2}, {2}} related to SCPKP ({1, 2, 3, 4}, 18)*

$$x_{1,1} + x_{1,3} + x_{2,2} + x_{3,2} + x_{4,2} \leq 2$$
$$x_{1,2} + x_{2,1} + x_{2,3} + x_{3,2} + x_{4,2} \leq 2$$

- *Pattern {{1, 2}, {1, 3}, {1, 3}, {1, 3}} related to SCPKP ({1, 2, 3, 4}, 18)*

$$x_{1,1} + x_{1,2} + x_{2,1} + x_{2,3} + x_{3,1} + x_{3,3} + x_{4,1} + x_{1,3} \leq 3$$
$$x_{1,1} + x_{1,3} + x_{2,1} + x_{2,2} + x_{3,1} + x_{3,3} + x_{4,1} + x_{1,3} \leq 3$$

The third set of inequalities does not contain any pattern inequalities of a characterizing SCPKP or ISCPKP.

$$x_{1,3} + x_{2,3} + x_{3,2} + x_{4,2} \leq 1$$

$$x_{1,2} + x_{2,2} + x_{3,1} + x_{3,3} + x_{4,1} + x_{4,3} \leq 2$$
$$x_{1,1} + x_{1,3} + x_{2,1} + x_{2,3} + x_{3,1} + x_{3,2} + x_{4,1} + x_{1,2} \leq 3$$

Note that for each inequality of the second set, permutations that are not facet-defining are dominated by an inequality of the third set. For instance, inequality $x_{1,3}+x_{2,3}+x_{3,2}+x_{4,3} \leq 1$ *is not present in the second set. This is because it is dominated by* $x_{1,3}+x_{2,3}+x_{3,2}+x_{4,2} \leq 1$ *present in the third set.*

In **Example 28**, three sets of binary inequalities are identified for the 1-HUC^D problem. The first two sets indicate that for a pattern-facet of a characterizing (I)SCPKP, there can be an associated GP yielding facet-defining inequalities of the 1-HUC^D problem. As the polyhedral symmetries only partially extend, and as shown in the second set, there are cases where not all inequalities associated with a GP are facet-defining. However, as the 1-HUC^D problem is more constrained than the (I)SCPKP, all permutations of the GP inequality yield valid inequalities of the 1-HUC^D problem. We also identified a third set of inequalities, which can be interpreted as reinforced GP inequalities. For this third set, permutations in the polyhedral symmetries of the 1-HUC^D problem yield facet-defining inequalities. However, any other permutation yields an invalid inequality. As such, we must distinguish two types of generalized patterns: GP and Reinforced GP (RGP). A GP is associated with a pattern of a characterizing (I)SCPKP, and a reinforced generalized pattern (RGP) is not associated with any pattern of a characterizing (I)SCPKP. For the GP, any permutation of the time periods yield valid inequalities. For the RGP, only permutations of time periods in a symmetric time set yield valid inequalities. Note that GP and RGP are complementary, as the former provides facet-defining inequalities that come directly from characterizing (I)SCPKP, whereas the latter provides facet-defining inequalities that do not come from characterizing (I)SCPKP. The aim is now to find how one can generate RGPs. For this purpose, we present **Example 29** illustrating how one can obtain an RGP from a GP. For the following example, we recall that conditions (i), (ii) and (iii) are defined in **Properties 8**, 9 and 10.

Example 29

Consider the same instance as in the previous ***Example 27****. The polyhedral symmetries of this instance accept permutations of time periods* $\{1,2\}$ *or* $\{3,4\}$.

Recall that $\mathcal{P} = \{\{2\},\{3\},\{3\},\{3\}\}$ *is a pattern of rank 1 from SCPKP ({1, 2, 3, 4}, 18). The associated GP is* $\mathcal{P} = \{\{2\},\{3\},\{3\},\{3\}\}$ *of rank 1, for time periods* $\{1,2,3,4\}$*. All pattern inequalities associated with GP* $\mathcal{P}$ *are valid for the 1-HUC^D problem. However, only the two following inequalities are facet-defining for the 1-HUC^D problem:*

$$x_{1,2} + x_{2,3} + x_{3,3} + x_{4,3} \leq 1$$
$$x_{1,3} + x_{2,2} + x_{3,3} + x_{4,3} \leq 1$$

We study one of the other permutations. Consider $\mathcal{X} = \{x_{1,3}, x_{2,3}, x_{3,3}, x_{4,2}\} \in \chi(\mathcal{P})$*. The associated GP inequality is:*

$$x_{1,3} + x_{2,3} + x_{3,3} + x_{4,2} \leq 1.$$

As $\mathcal{P}$ is pattern-facet for the SCPKP, then it is flexible pattern. As such, condition (iii) holds, meaning that there should be a solution with $x_{3,2} = 1$, $x_{3,3} = 0$ and $(pi(\mathcal{X}))$ to equality. Indeed, for the SCPKP, there is one such feasible solution, with $x_{3,1} = x_{3,2} = x_{4,1} = x_{4,2} = 1$ and all other variables to 0. However, such solution is infeasible for the 1-HUC^D problem, due to SCPKP ({3, 4}, 12). Therefore, condition (iii) does not hold for this particular permutation of GP $\mathcal{P}$. Hence, one can replace $x_{3,3}$ by $x_{3,2}$ for this GP inequality, keeping it valid and making it stronger. We obtain the following inequality:

$$x_{1,3} + x_{2,3} + x_{3,2} + x_{4,2} \leq 1$$

which is facet-defining for the 1-HUC^D. One can deduce the associated RGP $\mathcal{P}' = \{\{3\},\{3\},\{2\},\{2\}\}$. Besides, this RGP verifies conditions (i) (ii) and (iii) for this particular permutation.

In a same manner, one can obtain inequality $x_{1,2} + x_{2,2} + x_{3,1} + x_{3,3} + x_{4,1} + x_{4,3} \leq 2$ and its associated RGP from either pattern $\{\{1\},\{1\},\{2\},\{2\}\}$ or $\{\{1,3\},\{2\},\{2\},\{2\}\}$. Similarly, one can obtain inequality $x_{1,1} + x_{1,3} + x_{2,1} + x_{2,3} + x_{3,1} + x_{3,2} + x_{4,1} + x_{1,2} \leq 3$ and its associated RGP from pattern $\{\{1,2\},\{1,3\},\{1,3\},\{1,3\}\}$

Note however that a permutation of time periods for one of these RGP inequality yield an invalid inequality whenever this permutation is not supported by the polyhedral symmetry of this 1-HUC^D instance, i.e., permutations of time periods $\{1,2\}$ or $\{3,4\}$.

We can also generalize the definition of an (inverted) pattern-facet to (R)G(I)P. This definition differs from the one of a pattern-facet, as for an (R)G(I)P it is possible that only a fraction of the associated inequalities are facet-defining.

Definition 43 (*(Reinforced) generalized (inverted) pattern-facet*)

A (R)GP (resp. (R)GIP) $\mathcal{P}$ is facet-defining if there exists $\mathcal{X} \in \mathcal{X}(\mathcal{P})$ such that $(pi(\mathcal{X}))$ (resp. $(ipi(\mathcal{X}))$) is facet-defining.

We can also generalize conditions (i) (ii) and (iii), defined in **Properties 8**, 9 and 10, to obtain necessary conditions for an (R)GP to provide a facet-defining inequality. In the following, we detail three necessary facet-defining conditions (I), (II) and (III) for (R)GP. We do not detail how these conditions can be extended to the GIP and RGIP. This is because results for the (R)GP can directly apply to a (R)GIP. In the following we only detail results for the GP.

Condition (i), defined in **Property 8**, adapted to the (R)GP is exactly the same as for a pattern.

Property 32 (*Generalized condition* (I))

Let $\mathcal{P}$ be a GP defined for the time set $\mathcal{T}$. If $\mathcal{P}$ is a GP-facet, then it verifies conditions (I):

(I) *For every set $S_i(\mathcal{P}) \in \mathcal{P}$:* $|S_i(\mathcal{P})| \geq 1$

The proof is the same as the one for the condition (i) for the SCPKP in **Property 8**. We now generalize condition (ii), defined in **Property 9**, to a (R)GP defined on time set $\mathcal{T}$. In this case, the generalization

requires two modifications, in order to take the partial symmetry into account as well as all time periods in $\{1,\dots T\} \setminus \mathcal{T}$.

Property 33 (*Generalized condition (II)*)

Let $\mathcal{P}$ be a (R)GP of rank k defined on a time set $\mathcal{T}$. If $\mathcal{P}$ is (R)GP-facet, then $\mathcal{P}$ verifies condition (II) for at least one variable set $\mathcal{X} \in \chi(\mathcal{P})$:

(II) *For each $t \leq T$ there is $\mathcal{Y} \subseteq \mathcal{V}$ a k-intersection of $\mathcal{X}$ with $x_{tN} \in \mathcal{Y}$*

The following proof reuses the same ideas as to prove condition (ii) to be necessary in **Property 9**.

Proof : Let $\mathcal{P}$ be a (R)GP of rank k, and $\mathcal{X} \in \chi(\mathcal{P})$. Suppose there is a $t \leq T$ such that (II) is not satisfied for t. This means that there is no feasible solution with k variables of $\mathcal{X}$ set to 1 and in which $x_{tN} = 1$. Therefore, the following inequality is valid:

$$\sum_{x_{t'i'} \in \mathcal{X}} x_{t'i'} + x_{tN} \leq k$$

Indeed, when $x_{tN} = 0$, the inequality is valid by definition of the rank of $\mathcal{P}$. When $x_{tN} = 1$, the inequality is valid as there cannot be more than $k-1$ variables of $\mathcal{X}$ set to 1.

This inequality dominates the inequality $(pi(\mathcal{X}))$. Indeed, one could sum it with $-x_{tN} \leq 0$ to obtain $(pi(\mathcal{X}))$.

Thus, $(pi(\mathcal{X}))$ is facet-defining for the 1-HUC^D problem only if condition (II) is verified. Hence, $\mathcal{P}$ is (R)GP-facet only if there is $\mathcal{X} \in \chi(\mathcal{P})$ satisfying condition (II). ■

We now generalize condition (iii), defined in **Property 10**, to a (R)GP defined on a time set $\mathcal{T}$. In this case, the generalization requires one modification, in order to take into account the asymmetry of the bounds. But in opposition to (II), the following condition (III) does not require to take into account all time periods in $\{1,\dots,T\} \setminus \mathcal{T}$.

Property 34 (*Generalized condition (III)*)

Let $\mathcal{P}$ be a (R)GP of rank k defined on a time set $\mathcal{T}$. If $\mathcal{P}$ is (R)GP-facet for the 1-HUC^D problem, then $\exists \mathcal{X} \in \chi(\mathcal{P})$ that verifies condition (III):

(III) *For each variable $x_{ij} \in \mathcal{X}$, there is $\mathcal{Y} \subseteq \mathcal{V}$ a k-intersection of $\mathcal{X}$ with $x_{ti-1} \in \mathcal{Y}$ and $x_{ti'} \notin \mathcal{Y}$ $\forall i' \geq i$.*

The following proof reuses the same ideas as to prove condition (iii) to be necessary in **Property 10**.

Proof : Let $\mathcal{P}$ be a (R)GP of rank k. Let $\mathcal{X} \in \chi(\mathcal{P})$. Suppose for some $x_{ti} \in \mathcal{X}$ does not verify condition (iii). This means that there is no feasible solutions with a total of k variables of $\mathcal{X}$ set to 1, with $x_{ti-1} = 1$ and $x_{ti} = 0$. Therefore, the following inequality is valid:

$$\sum_{x_{t'i'} \in \mathcal{X}} x_{t'i'} + x_{ti-1} - x_{ti} \leq k$$

Indeed, when $x_{ti} = x_{ti-1} = 1$ or $x_{ti} = x_{ti-1} = 0$, this inequality is valid by the rank of $\mathcal{P}$. When $x_{ti-1} = 1$ and $x_{ti} = 0$, the inequality is valid as there cannot be more than $k-1$ variables of $\mathcal{X}$ equal to 1, which sums to at most k. Case $x_{ti-1} = 0$ and $x_{ti} = 1$ does not exist due to the order constraints

This inequality dominates the inequality $(pi(\mathcal{X}))$. Indeed, one could sum it with $-x_{ti-1} + x_{ti} \leq 0$ (equivalent to $x_{ti} \leq x_{ti-1}$) to obtain $(pi(\mathcal{X}))$.

Thus, $(pi(\mathcal{X}))$ is facet-defining for the 1-HUC^D problem only if condition (iii) is verified. Hence, $\mathcal{P}$ is (R)GP-facet only if $\exists \mathcal{X} \in \chi(\mathcal{P})$, which verifies condition (iii). ■

As the polyhedral symmetries do not hold completly for the 1-HUC^D problem, if a pattern satisfies (ii) and (iii) for an SCPKP, the associated GP does not necessarily satisfy (II) and (III). Note however that if a pattern satisfies (i) for an SCPKP, the associated GP automatically satisfies (I) for the 1-HUC^D problem, as conditions (i) and (I) are similar. In the following, we detail the idea of an extention of the two-phase Branch & Cut algorithm defined in **Chapter 4** to the 1-HUC^D problem.

5.6 Generalization of the two-phase Branch & Cut algorithm

In this section, we extend the two-phase B&C algorithm defined for the SCPKP in **Chapter 4** to the 1-HUC^D problem. The main idea is to add pre-processing steps, in order to first identify the characterizing defining-(I)SCPKPs, and then the characterizing induced-(I)SCPKPs. Once they are identified, one can reuse the two-phase algorithm, with patterns from each of these characterizing (I)SCPKPs.

Identifying the characterizing defining-(I)SCPKPs The defining-(I)SCPKPs can be directly obtained with constraints β_t^* and α_t^*. In order to have the characterizing defining-(I)SCPKPs, we compare the defining-(I)SCPKPs pairwise. Indeed, consider two defining-SCPKPs, respectively $(\mathcal{T}, C)$ and $(\mathcal{T}', C')$. If $\mathcal{T} \subset \mathcal{T}'$ and $C \leq C' - |\mathcal{T}' \setminus \mathcal{T}| \sum_{i=1}^{N} D_i$, then SCPKP $(\mathcal{T}', C')$ is not characterizing. This is because any SCPKP $(\mathcal{T}'', C'')$ on time set $\mathcal{T}''$ with $C'' = |\mathcal{T}''| \sum_{i=1}^{N} D_i$ is a sub-problem of 1-HUC^D. Hence, with $\mathcal{T}'' = \mathcal{T}' \setminus \mathcal{T}$, the SCPKPs $(\mathcal{T}, C)$ and $(\mathcal{T}'', C'')$ dominate SCPKP $(\mathcal{T}', C')$ as shown in **Property 25**. In a similar fashion, consider two defining-ISCPKPs, respectively $(\mathcal{T}, C)$ and $(\mathcal{T}', C')$. If $\mathcal{T} \supset \mathcal{T}'$ and $C \geq C' + |\mathcal{T} \setminus \mathcal{T}'| \sum_{i=1}^{N} D_i$, then ISCPKP $(\mathcal{T}', C')$ is dominated.

Obtaining characterizing induced-(I)SCPKPs As shown in **Property 30**, the bounds of the characterizing defining-(I)SCPKPs must be tightened in order for the induced (I)SCPKP not to be underconstrained. Hence, we first tighten the bounds of the characterizing defining-(I)SCPKPs by solving the MILP model defined in **Section 5.4**. Then, characterizing induced (I)SCPKPs can be obtained from each pair of one defining-SCPKP and one defining-ISCPKP. Recall that there are exactly T defining-SCPKPs and defining-ISCPKPs, meaning that it requires at most T^2 comparisons. From **Properties 28** and **29**, there cannot be any other characterizing (I)SCPKP than the ones obtained.

Extending the two-phase B&C Once the characterizing (I)SCPKPs have been obtained, the first phase of the two-phase B&C would be to generate patterns for all of these (I)SCPKPs. In this case, one needs to verify condition (I), (II) and (III), but in practice it is very similar to verifying (i), (ii) and (iii). This

can be done in parallel for different (I)SCPKPs, as the pattern generation procedure for an (I)SCPKP is independent of the procedure for another (I)SCPKP. Finally, for the 1-HUC^D problem, the second phase would be also very similar to that for an SCPKP.

Within the alloted time for this thesis, we could not make an extensive numerical comparisons of the generalized version of the two-phase B&C algorithm with state-of-the-art algorithms. However, very preliminary results have shown that such an algorithm could be more efficient than default CPLEX.

5.7 Conclusion

In this chapter, we extended the polyhedral study of **Chapter 4**, dedicated to the Symmetric-weight Chain Precedence Knapsack Problem (SCPKP), to the discretized 1-Hydro Unit Commitment problem without ramping nor min-up/down constraints (1-HUC^D). For this purpose, we first defined the Inverted SCPKP (ISCPKP), similar to the SCPKP but for which the knapsack inequality that sets an upper bound on the weight of the selected items is replaced by a covering inequality, defining a lower bound on the weight of the selected items. We indicated that an ISCPKP can cast in an SCPKP, meaning that polyhedral results of the latter translate to the former. Then, we showed that the constraints of the 1-HUC^D problem is the union of the constraints of multiple (I)SCPKPs. In addition, we proved that only a polynomial number of (I)SCPKPs are relevant to study the polyhedron associated to the 1-HUC^D. As such, it makes it suitable to use polyhedral results of the (I)SCPKP to obtain strong inequalities for the 1-HUC^D problem. Even if the polyhedral symmetries do not appear systematically for the 1-HUC^D problem, we did characterize sets of time periods for which polyhedral symmetries arise. Thus, we generalized the patterns to handle these cases, as well as three necessary facet-defining conditions. Then, we adapted the two-phase Branch & Cut algorithm defined in **Chapter 4**. For this purpose, we presented a few complementary pre-processing steps in order to obtain the characterizing (I)SCPKP, from which the two-phase B&C algorithm can be used.

In the following chapter, we propose graph algorithms to solve the generalization of the 1-HUC^D problem, namely the 1-HUC^{DRM} problem, as an alternative approach to solving a MILP model.

Part III

Graph algorithms

6 Graph algorithms for the discretized 1-Hydro Unit Commitment problem

Table of contents

In the previous chapter, we presented a polyhedral study for a linear model of the 1-Hydro Unit Commitment (1-HUC) problem, to enhance the current approach at EDF. However, this study does not account for some constraints of the 1-HUC problem, namely ramping and min-up/down constraints. Besides, the discretized 1-HUC problem has a very specific structure, which makes it possible to represent it with graphs. Hence, graph algorithms can yield competitive approaches to solving a MILP model of the 1-HUC problem.

In this chapter, we propose two graph algorithms and their corresponding graph representation to solve a discretized 1-HUC problem taking ramping and min-up/down constraints into account. In **Section 6.1** we define the considered problem, as well as a bi-objective problem used in one of the algorithms proposed. In **Section 6.2** we introduce two graph representations of the considered 1-HUC problem. In **Section 6.3** we review graph algorithms for related problems. In **Section 6.4** we present a first graph algorithm. In **Section 6.5** we detail a second graph algorithm. The numerical results for each algorithm are in their corresponding section. In **Section 6.6**, we draw concluding remarks. The first graph algorithm has been published in [52].

6.1 Problems definition

The 1-HUC problem considered in this chapter is the 1-HUC^{DRM} problem, defined in **Section 3.3**.

We also give the definition of the Shortest Path Problem with Resource Windows (RWSPP) since we will show in **Section 6.2** how the 1-HUC^{DRM} problem can be defined as a special case of the RWSPP. Let $G = (V, A)$ be a graph, with V the set of vertices and A the set of arcs. We denote s and p the source vertex and the target vertex, respectively. Each arc $a \in A$ has a value $V(a) \in \mathbb{R}$ and a resource amount $R(a) \in \mathbb{R}_{\geq 0}$. Each vertex $v \in V$ has a resource window $[\underline{R}(v); \overline{R}(v)]$. For a path C, we denote $V(C) = \sum_{a \in C} V(a)$ the value of the path, and $R(C) = \sum_{a \in C} R(a)$ the amount of resource used. A path C from s to v is locally feasible if $R(C) \in [\underline{R}(v); \overline{R}(v)]$. A path C is feasible if all sub-paths C' of C starting from s are locally feasible. The RWSPP consists in finding feasible path C from s to p minimizing $V(C)$. Clearly, the RWSPP is a generalization of the Shortest Path Problem with Resource Constraints (RCSPP), as the latter only considers an upper bound on the resource.

For the purpose of one of the algorithms proposed, we also defined the Bi-Objective Shortest Path Problem (BOSPP), as follows. Let $G = (V, A)$ be a graph, with V the set of vertices and A the set of arcs. We denote s and p the source vertex and the target vertex, respectively. Each arc $a \in A$ has two values $V_1(a) \in \mathbb{R}$ and $V_2(a) \in \mathbb{R}$. For a path C, we denote $V_i(C) = \sum_{a \in C} V_i(a)$ the value for objective $i \in \{1, 2\}$. The BOSPP consists in finding a set of Pareto-optimal paths C from s to p minimizing $V_1(C)$ and $V_2(C)$. In the case of bi-objective optimization, two solutions can be incomparable, meaning that the standard definition of optimal solution does not hold. As such, we recall the definition of a Pareto-optimal path.

Definition 44 (*Pareto-optimal paths*)

> *A path C is Pareto-optimal if there is no other path C' such that for each $i \in \{1, 2\}$, $V_i(C') \geq V_i(C)$ and there is $i \in \{1, 2\}$ for which $V_i(C') > V_i(C)$.*

We also recall the definition of Pareto-supported paths for the purpose of one of the algorithms proposed.

Definition 45 (*Pareto-supported paths*)

> *A path C is Pareto-supported if it is Pareto-optimal and is in the solutions' convex hull in the objective*

space.

For readability purposes, we also introduce an operator indicating that two paths have the same value for both objectives and its opposite operator.

Definition 46 (*Pareto-equality*)

We define the Pareto-equality $=_p$ *such that for two paths* C_1 *and* C_2, $C_1 =_p C_2$ *if and only if* $V_1(C_1) = V_1(C_2)$ *and* $V_2(C_1) = V_2(C_2)$.

Definition 47 (*Pareto-inequality*)

We define the Pareto-inequality $\neq_p$ *such that for two paths* C_1 *and* C_2, $C_1 \neq_p C_2$ *if and only if* $V_1(C_1) \neq V_1(C_2)$ *or* $V_2(C_1) \neq V_2(C_2)$

6.2 Graph representations of the 1-Hydro Unit Commitment problem

In this section, we show two graph representations of the 1-HUC^{DRM} problem. The first one is a cumulated flow-expanded ($\mathcal{D}_{1,t}$-expanded) graph, where the cumulated flow $\mathcal{D}_{1,t}$, defined in **Section 3.2** is represented in each vertex. The second one is a compact graph, where $\mathcal{D}_{1,t}$ is a resource. The $\mathcal{D}_{1,t}$-expanded one makes it possible to account for the resource windows by discarding vertices whenever the resource does not comply with the window. In the following we describe both representations and their associated dominance rules.

6.2.1 The cumulated flow-expanded graph

Let $G_E = (V_E, A_E)$ denote the graph defined as follows. Each vertex $u \in V_E$ is defined as a quadruplet $u = (t, i, d, l)$. For this definition, t is the time period, i is the operating point, d is the cumulated flow $\mathcal{D}_{1,t}$ associated with a path that reaches u, $l \in \{-L+1, \ldots, L-1\}$ is the remaining time for min-up/down constraints to be satisfied. In the case $l > 0$ (resp. $l < 0$), then l indicates the number of time periods starting from t during which the flow cannot decrease (resp. increase) in order to satisfy min-up (resp. min-down) constraints. The vertices will be illustrated in **Example 30**. Without loss of generality, $u = (t, i, d, l)$ is considered only if $d \in [\beta_t^*; \alpha_t^*]$. The source vertex s is defined as $s = (0, 0, 0, 0)$ and the target vertex p as $p = (T+1, 0, 0, 0)$.

There is an arc $a \in A_E$ from each vertex (T, i, d, l) towards p of value 0. For the following description of the arcs, consider a vertex $u = (t, i, d, l) \in V_E$ with $t \in \{0, \ldots, T-1\}$.

If $l \geq 1$, there is an arc $a \in A_E$ from u to $(t+1, i, d + \sum_{j=0}^{i} D_j, l-1)$. Similarly, if $l \leq -1$, there is an arc $a \in A_E$ from u to $(t+1, i, d + \sum_{j=0}^{i} D_j, l+1)$. If $l \geq 0$, for any $i' > i$ such that $\sum_{j=i+1}^{i'} D_j \leq R_u$ there is an arc $a \in A_E$ from u to $(t+1, d + \sum_{j=0}^{i'} D_j, L-1)$. Similarly, if $l \leq 0$, for any $i' < i$ such that $\sum_{j=i'+1}^{i} D_j \leq R_d$ there is an arc $a \in A_E$ from u to $(t+1, d + \sum_{j=0}^{i'} D_j, -L+1)$.

The value of any arc towards a vertex (t, i, d, l), with $t \leq T$, is $\sum_{j=0}^{i} \Psi_{t,j}$.

Solving a Longest Path Problem (LPP) on G_E provides an optimal solution of the 1-HUC^{DRM} problem. Indeed, by construction, any path in G_E satisfies min-up/down, ramping and resource windows. Besides, the value of any path in G_E is exactly the value of the corresponding 1-HUC^{DRM} problem's solution.

The downside of such a graph is its pseudo-polynomial number of vertices, as proven in the following property.

Property 35 (*Pseudo-polynomial number of vertices of G_E*)

The number of vertices in G_E is pseudo-polynomial with respect to the size of the instance.

Proof : As we consider that there are no pump, then for any $t < T$, $\beta_t^* \leq \beta_{t+1}^*$ and $0 \leq \alpha_t^* \leq \alpha_{t+1}^*$. Hence, at each time period, the cumulated flow $\mathcal{D}_{1,t}$ lies in $[0; \alpha_T^*]$. As $l \in \{-L+1, \ldots, L-1\}$ and $i \in \{0, N\}$, there are at most $(\alpha_T^*+1)\cdot(N+1)\cdot(2L-1)$ vertices per time period. This yield a total of $(\alpha_T^*+1)\cdot T\cdot(N+1)\cdot(2L-1)+2$ vertices, taking into account the source and target vertex. As $L \leq T$, we deduce that the number of vertices is at most $(\alpha_T^*+1)\cdot T\cdot(N+1)\cdot(2T-1)+2$.

From the definition of the 1-HUC^{DRM} problem in **Section 3.3**, there are data for each operating point as well as for each time period. Hence the size of the instance is of $\mathcal{O}(T \cdot N)$. The number of vertices is pseudo-polynomial as α_T^* is not polynomially bounded with respect to the size of the instance. ■

However, we can use classical dominance rules for the longest path:

Definition 48 (*Dominance rule 1*)

Let p and q be two paths from s to a vertex u. By induction the path with the lowest value is dominated, as it cannot lead to an optimal solution.

Definition 49 (*Dominance rule 2*)

Let p be a path from s to u going through a vertex v, and q be a path from s to v. Let $p_{s,v}$ be the subpath of p from s to v and $p_{v,u}$ the subpath of p from v to u. If the value of $p_{s,v}$ is larger than that of q, then q is dominated. If the value of q is larger than that of $p_{s,v}$, then p is dominated by the path concatenating q and $p_{v,u}$.

By construction of G_E, any vertex $(t, 0, d, l)$ with $l \geq 1$ cannot be reached. This is because by construction, an arc heading to $(t, 0, d, l)$ with $l \geq 1$ can only exist from a vertex $(t-1, i, d', l')$ with $i < 0$, which does not exist. We can explain in the same manner that vertices $(t, N, d\ l)$ with $l \leq -1$ cannot be reached either. Hence, it is not necessary to consider these vertices in G_C.

Example 30

Consider an instance of the 1-HUC^{DRM} problem for $T = 5$ time periods. Accounting for the idle operating point, the unit operates on 3 operating points: ($D_0 = 0$, $P_0 = 0$), ($D_1 = 6$, $P_1 = 8$), ($D_2 = 5$, $P_2 = 6$). The ramps rates are $R_u = R_d = 6$ and the duration of the min-up/down constraints is $L = 3$. The bounds are β^* =[0, 0, 7, 18] and α^* =[11, 18, 18, 18].

The value of operating points are: $\Psi_{i,1} = [0, 2.8, 1]$, $\Psi_{i,2} = [0, -6.8, -6.2]$, $\Psi_{i,3} = [0, 0.4, -0.8]$, $\Psi_{i,4} = [0, -11.6, -9.8]$ *and* $\Psi_{i,5} = [0, 2.0, 0.4]$.

Figure 6.1 *depicts the graph* G_E *associated with this instance of the 1-HUC*^{DRM} *problem. For readability purposes, only vertices that can be reached from* s *are represented. Dotted lines separate the vertices with respect to the cumulated flow. The values of the arcs are not represented in this graph. As the value and the resource only depend on the time period* t *and the operating point* i *they are the same for any arc heading to any vertex* (t, i, d, l) *for a given* t *and* i*. In* ***Table 6.1****, we give the value for each combination of* $t \in \{1, \dots, T\}$ *and* $i \in \{0, \dots, N\}$.

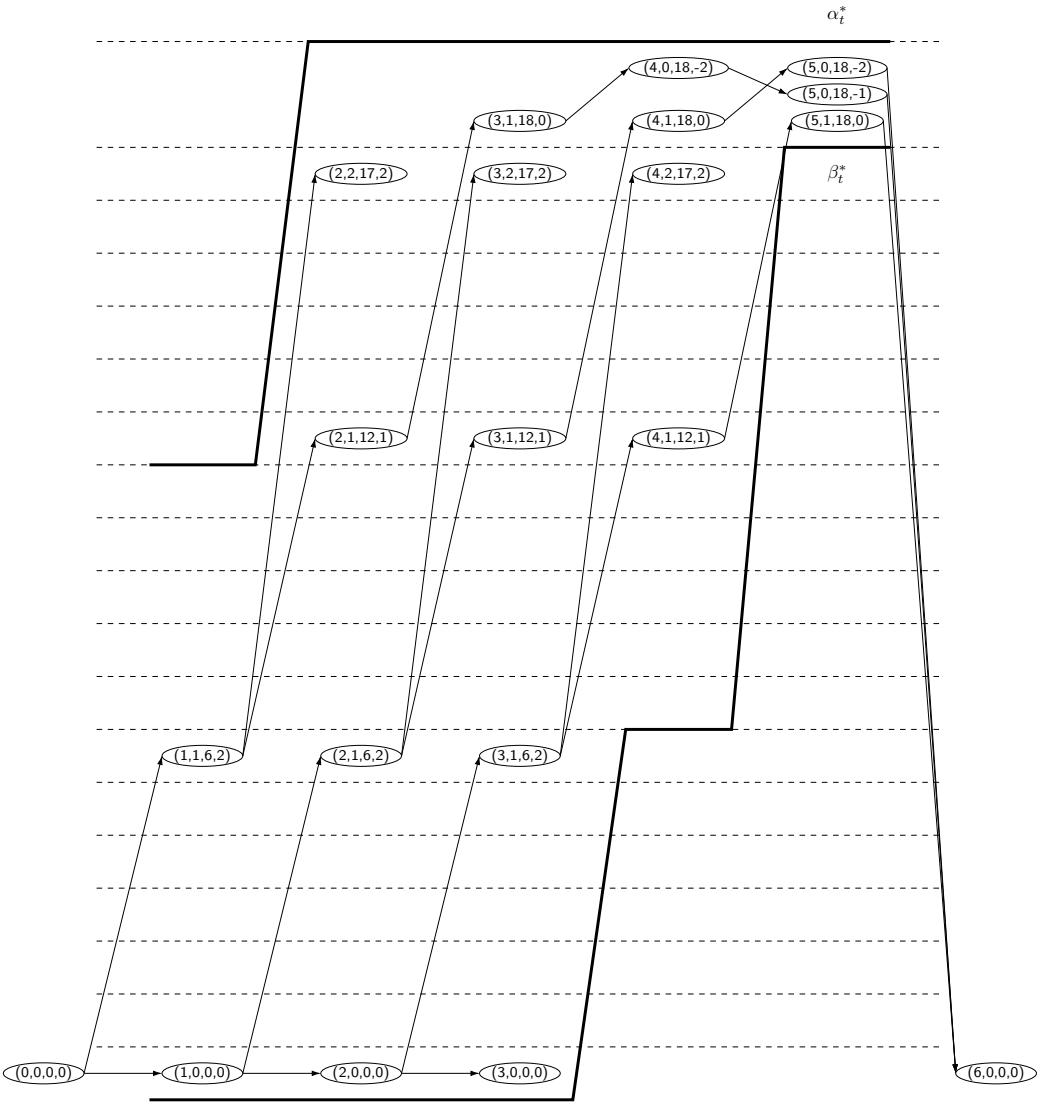


Figure 6.1: Graph G_E of **Example 30**

	$i=0$	$i=1$	$i=2$
$t=1$	0	2.8	3.8
$t=2$	0	-6.8	-13.0
$t=3$	0	0.4	-0.4
$t=4$	0	-11.6	-21.4
$t=5$	0	2.0	2.4

Table 6.1: Values of the arcs of graph G_E depicted in **Figure 6.1**

Remark 7

The 1-HUC^{DRM} problem is solved in practice on a daily basis, and the plant is not stopped between two consecutive days. This means that one needs to consider the last decision of the previous day when solving the 1-HUC^{DRM} problem. This can be taken into account by modifying source vertex s as follows. Consider the path for the previous day going through vertex (T,i,d,l). One can initialize $s=(0,i,0,l)$. As such, the plant can be operating continuously from one day to another, without violating any constraint. Note however that modifying vertex s in such a manner does not modify the structure of the graph. Hence, in the following we consider s=(0,0,0,0) without loss of generality.

6.2.2 The compact graph

The graph representation proposed in this section is an extension of the RCSPP representation of [3], as we take into account min-up/down constraints. The resource considered is the cumulated flow $\mathcal{D}_{1,t}$.

Let $G_C=(V_C,A_C)$ be the graph defined as follows. Each vertex $u\in V_C$ is defined as a triplet (t, i, l). For this definition, t, i and l have the exact same definition as for $\mathcal{D}_{1,t}$-expanded graph G_E. Also, a source vertex and target vertex, respectively s=(0,0,0) and p=$(T+1,0,0)$ are defined. For any vertex $u=(t,i,l)\in V_C\setminus\{s,p\}$ we define the resource window $[\underline{R}(u)=\beta_t^*;\overline{R}(u)=\alpha_t^*]$. For vertex s, the resource window is $[\underline{R}(s)=0;\overline{R}(s)=\infty]$ and for p it is $[\underline{R}(p)=\beta_T^*;\overline{R}(p)=\alpha_T^*]$.

There is an arc in A_C from each vertex (T, i, l) towards p of value 0 and which uses 0 resource. For the following description of the arcs, consider a vertex u=(t, i, l) with $t\in\{0,\dots,T-1\}$.

If $l\geq 1$ there is an arc $a\in A_C$ towards $(t+1,\,i,\,l-1)$. Similarly, if $l\leq -1$ there is an arc $a\in A_C$ towards $(t+1,\,i,\,l+1)$. If $l=0$, there is an arc towards $a\in A_C$ $(t+1,\,i,\,0)$

If $l\geq 0$, for any $i'>i$ such that $\sum_{j=i+1}^{i'}D_i\leq R_u$ there is an arc $a\in A_C$ from u to $(t+1,\,i',\,L-1)$, or $(t+1,\,i',0)$. Similarly, if $l\leq 0$, for any $i'<i$ such that $\sum_{j=i+1}^{i'}D_i\leq R_d$ there is an arc $a\in A_C$ from u to $(t+1,i',-L+1)$.

The value of any arc towards a vertex $(t,\,i,\,l)$, with $t\leq T$ is $\sum_{j=0}^{i}\Psi_{t,j}$, and the resource used is $\sum_{j=1}^{i}D_j$.

The downside of G_C, is that there are paths in this graph which do not satisfy the resource constraints. Hence, we need to verify for each path considered if the resource windows are satisfied. As such, solving the RWLPP on G_C provides an optimal solution of the 1-HUC^{DRM} problem. The idea is similar to the one used to show that solving the LPP on G_E provides an optimal solution of the 1-HUC^{DRM} problem.

The only difference is that the resource windows are not explicitly addressed by the graph, hence we aim to solve an RWLPP on G_C.

On the positive side, the number of vertices is polynomial.

Property 36 (*Polynomial number of vertices of G_C*)

The number of vertices of G_C is polynomial with respect to the size of the instance.

Proof : Indeed, there are $(N+1)\cdot(2L-1)$ vertices per time period. Hence, there are $T\cdot(N+1)\cdot(2L-1)+2$ vertices in G_C in total.

Similarly to the proof of **Property 35**, the size of the instance is of $\mathcal{O}(T\cdot N)$. As such, the number of vertices of G_C is polynomial with respect to the size of the instance. ■

Also, it is possible to use a classical dominance rule for the RCSPP under certain conditions, as defined in [3]:

Definition 50 (*Dominance rule 3*)

Consider two partial paths C_1 and C_2, both from the source vertex s to a vertex u. Consider the case where $\underline{R}(u) \leq R(C_2)$ for all $u \in V_C$ and $R(C_1) \leq R(C_2)$. For any path completion C_3 from u to the target vertex p such that (C_2, C_3) form a feasible path, then (C_1, C_3) also form a feasible path. Consequently, if $V(C_1) \geq V(C_2)$ then C_2 is dominated.

Note that the condition on the resource usage to ensure that the lower bounding constraints (3.2.6) are satisfied seriously weakens the dominance rules when these constraints are active.

By construction of G_C, any vertex $(t, 0, l)$ with $l \geq 1$ or (t, N, l) with $l \leq -1$ cannot be reached. The reason is the same as for why there is no arc towards $(t, 0, d, l)$ with $l \geq 1$ or (t, N, d, l) with $l \leq -1$ in G_E.

Example 31

*Consider the instance of **Example 30**. **Figure 6.2** represents the graph G_C associated to this instance. The vertices appear in black if they can be reached from s, and in gray otherwise. Dotted lines separate the vertices with respect to each operating point. For readability purposes, the values and the resource of the arcs are not represented in this graph. As the value only depends on the time period t and the operating point i they are the same for any arc heading to any vertex (t, i, l) no matter the value of l. **Table 6.1** gives the value of each combination of t and i. Similarly, the resource only depends on the operating point i they are the same for any arc heading to any vertex (t, i, l) no matter the value of t or l. For operating point 0, the resource is 0; for operating point 1, the resource is 6 and for operating points 2, the resource is 11.*

A similar remark as **Remark 7** can be made for G_C Indeed, one can take into account the decision at time period T of the previous day by modifying source vertex s. This does not change the structure of the graph, hence we consider in the following $s = (0, 0, 0)$ without loss of generality.

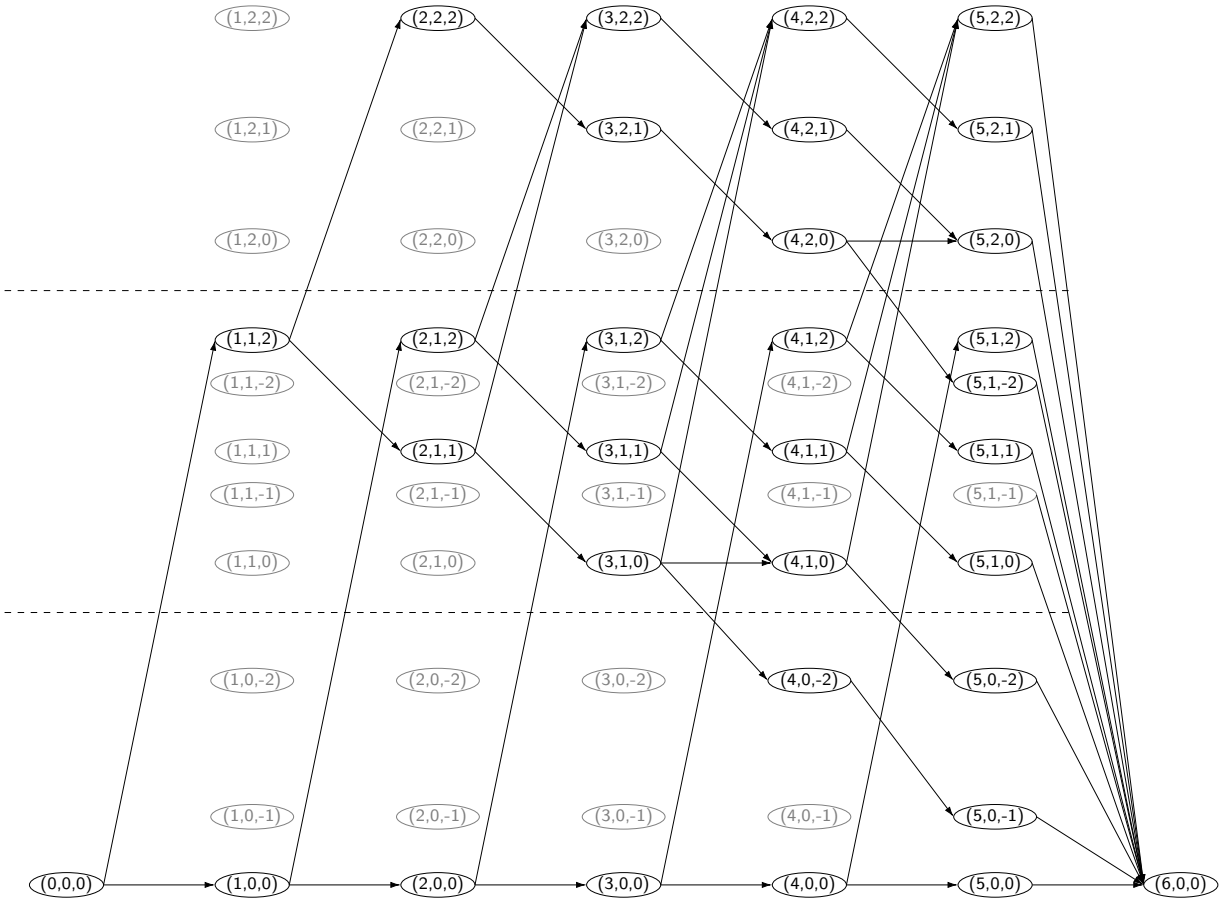


Figure 6.2: Graph G_C of **Example 31**

Property 37

Any path in graph G_C satisfies ramping constraints and min-up/down constraints.

Proof : Consider an arc between (t, i, l) and $(t+1, i', l')$. Clearly, in the case $i = i'$, the ramping constraints are satisfied if such an arc exists in G_C. In the case $i < i'$, such an arc only exists if $\sum_{j=i+1}^{i} D_j \leq R_d$ by construction of G_C. In the case $i > i'$, such an arc only exists if $\sum_{j=i'}^{i-1} D_j \leq R_u$ by construction of G_C. Consequently, ramping constraints are always satisfied.

By construction of G_C, from a vertex (t, i, l), the only way to reach operating point $i' > i$ is through the arc towards $(t+1, i', L-1)$. Also by construction of G_C, from $(t+1, i', L-1$,u), one needs to go through all vertices $(t+1+\tau, i', L-1-\tau)$, with $\tau \in \{1, \ldots, L-2\}$, and $(t+L, i', 0)$ in order to reach operating point $i'' < i'$. In total, a path must go through at least $1 + L - 2 + 1 = L$ vertices with operating point i' before heading to a vertex with operating point $i'' < i'$. Hence, the min-up constraint is satisfied. We can prove in a similar way that the min-down constraints are satisfied. ■

The proof of **Property 37** is illustrated by **Figure 6.2** from **Example 31**.

Remark 8

Note that for both representations, the associated graph is acyclic. For acyclic graphs, the SPP and the LPP and their variants are equivalent. Hence, despite the fact that the 1-HUC^{DRM} problem is a maximization problem, algorithms for the SPP and its variant still can be used. As the use case 1-HUC^{DRM} problem is a maximization problem, our algorithms will be described for variants of the LPP.

6.3 Literature review

In this section, we first present a literature review of graph algorithms developed for the UCP and for the HUC. As we consider the HUC an an RWSPP, we also include in this review graph algorithms for the RWSPP. As graph algorithms are essentially dynamic programming algorithms, we focus on such a type of algorithms.

6.3.1 Dynamic programming for the Unit Commitment Problem

A dynamic programming algorithm for a single plant Unit Commitment (1-UC) problem with ramping and min up/down constraints is presented in [38]. The algorithm is based on a graph with a source vertex and several groups of T vertices. For each even (resp. odd) group, vertex t indicates that the unit is turned off (resp. on) at time period t. The arcs connect the vertices of a group to the next groups, from a time period t to a time period $t' > t$. Finding a path in this graph allows one to find an on-off schedule for the units of the plant. The difference between the 1-UC and the 1-HUC^{DRM} problem is that when the 1-UC problem features a resource, it has an upper bound [60], while in the 1-HUC^{DRM} problem, the presence of reservoirs with minimum and maximum volumes requires to account for resource windows.

6.3.2 Dynamic programming for the Hydro Unit Commitment problem

In this part we focus on dynamic programming algorithms for the HUC problem, most of them being cited in the survey [97].

In [87] the author presents a two-phase approach to solve the HUC problem, as an LP for the first phase and using a dynamic programming algorithm for the second phase. More precisely, the second phase consists in solving a 1-HUC problem for each plant of the valley with dynamic programming, aiming to get the closest solution to the LP solution while taking into account constraints omitted in the LP. For this phase, the considered underlying graph is similar to G_E, but the reservoir volume is discretized into hundreds of possible volume values for a reservoir. Consequently, there are hundreds of vertices per time period. A Bellman-Ford algorithm [11] is used to find a path in this graph. Such a discretization discards a lot of realistic states from G_E. In our study, we consider all possible states in

G_E, which is pseudo-polynomial. This can yield a very large number of states, specially for instances with a large number of time periods. With such a large number of states, the Bellman-Ford algorithm becomes far less efficient.

In [82] a method for solving a non-linear 1-HUC problem with a target volume is described. To solve this problem with dynamic programming, a state diagram is constructed. In a similar fashion to [87], evenly discretized volumes are considered, yielding a limited number of states per time periods. More precisely, it is stated in the experimental results of [82] that the discretization is ranging from 0.3% to 0.5% of the difference between the minimum and maximum volume of the reservoir. This represents about 300 vertices at each time period, which discards the majority of the states of G_E when taking into account all possible states. In order to have feasible solutions, the target volume is relaxed to match this discretization. The state diagram is constructed by generating the possibilities to reach the target volume from the initial volume, satisfying the upper and lower bounds on the volume at each time period. Starting from the state at the end of the time horizon, the dynamic programming algorithm maximizes the value of the generated power. As we consider instances of the 1-HUC^{DRM} problem with and without target volumes, a backward algorithm may not be practical with large volumes and no target volume.

In [3], a decomposition method for solving the HUC problem with shortest paths is described. The considered HUC problem is relative to a valley where each plant has a finite number of operating points. The topology of the valley is not restricted to a chain, as each plant (resp. reservoir) can have a set of upstream and downstream reservoirs (resp. plants). This HUC problem also takes into account ramping constraints as well as a target volume for each reservoir at the last time period. Note that the latter target volume is a minimal bound, meaning there is no equality constraint. The solution approach decomposes this HUC problem into multiple 1-HUC problems. Each 1-HUC problem is represented by a graph similar to G_C. The 1-HUC subproblems without resource constraints are solved by a shortest path algorithm, while the ones with resource constraints are solved by a labeling algorithm defined in [2]. The latter algorithm is adapted from a classical RCSPP algorithm [9] to take into account a minimum bound for the resource. It is mentioned that this labeling algorithm loses its dominance properties between two labels if one of them does not verify the minimum bound on the resource. Such a case is more frequent when the target volume is considered with equality constraints, making this algorithm much less efficient.

There are also other problems solved by dynamic programming related to the 1-HUC problem. In [5] a dynamic programming approach is described to solve an HUC problem on instances of the Itaipù plant (Brazil, Paraguay). This problem differs from ours as the only constraint is to satisfy the minimum and maximum number of turbines running at each time period. No volume is considered, therefore there are neither bounds on the volume, nor a target volume. In [23] the Hydro Unit Load Dispatch problem (HULD) is presented. This problem differs from the HUC problem, as the water flow is known, and solving the HULD problem is to provide the most economic distribution of the water through the different turbines, while verifying the flow capacity of the turbines at each time period.

6.3.3 Shortest path with resource constraints

As the HUC can be modeled an an RWSPP we are interested in the solution methods for the RCSPP and the RWSPP.

There are works on the RCSPP to solve the thermal problem on EDF instances [60]. In that paper it is indicated that the resource has an upper bound but no lower bound. However, as specified in [3], the difficulty of the HUC comes from the lower bound on the volume, which weakens the dominance rules.

In [102], a state-of-the-art review of different shortest path variants is described. More specifically, it is indicated that there is little work on the RWSPP. Three papers are cited, namely [88] describing a heuristic, [10] presenting an integer formulation and [109] proposing a dynamic programming algorithm. As we look for an exact algorithm and an alternative to integer programming, we will focus on the three-phase algorithm described in [109]. The presented algorithm solves the RWSPP on acyclic graphs. The main idea, further detailed in [110], is to extend the graph, such that if multiple paths lead to the same vertex from the source, a new vertex is created for each of these paths. The problem is then solved on the expanded graph. However, the expanded graph is of exponential size, making it difficult to use in practice. In the case of the 1-HUC^{DRM} problem, such an extension can lead up to N^T vertices at time period T.

In [56], efficient dominance rules for the Shortest Path Problem with Time Windows and Time Cost (SPPTWTC) exist when there is also a time-cost, meaning more generally when the objective function is directly linked to the resource. Recall that as explained in **Section 3.2**, time windows are less restrictive than resource windows due to the possibility of waiting at a vertex for the window opening. Hence, these dominance rules do not exist as such for the RWSPP. Nevertheless, one can try to extend these dominance rules to the RWSPP in the case where there is a resource-cost. However, the 1-HUC^{DRM} problem does not fall into this family of RWSPP. Indeed, the objective function depends on both the flow D_t and the power P_t, the latter not being linked to the resource, namely the cumulated flow $\mathcal{D}_{1,t}$.

6.3.4 Bi-objective approaches

The two-phase method was first introduced to solve the bi-objective knapsack problem [106]. This classical method is particularly efficient to deal with two or even three objectives when the corresponding problem with a single objective is easy to solve. In the bi-objective case, the two-phase method can be defined as follows. The first phase consists in obtaining one Pareto-supported solution for each Pareto-supported point that form the convex envelope of the objectives' space. This is because if Pareto-supported solutions are known, then one can drastically reduce the search space to obtain Pareto-optimal solutions, i.e., solutions that are never dominated on all objectives by another solution (see **Section 6.1** for definitions). The main idea to obtain Pareto-supported solutions is to solve the problem with the two objectives aggregated into a single one by convex combination. The set of Pareto-supported solutions is generated by varying the combination parameters. The second phase is to use an enumeration algorithm, to obtain the remaining Pareto-optimal solutions within the reduced search

space defined by the Pareto-supported solutions. The most straightforward approach is the enumeration of one Pareto-optimal solution for each Pareto-optimal point in the objectives' space, which can be done with a Branch & Bound variant [106] or dedicated algorithms [35]. As the number of Pareto-optimal solutions can be exponential, such algorithms can become very time-consuming, and many other approaches have been developed.

Whenever it is possible to interact with the decision maker, one can consider an interactive algorithm [72]. The main principle is to iteratively suggest solutions to the decision maker, who progressively provides preference information. Such an approach can drastically reduce the search space at each iteration. The main downside is that the decision maker must be available during the execution of the algorithm, which is not the case for the 1-HUC^{DRM} problem.

When no interaction with the decision maker is possible, the computational time is linked to the number of Pareto-optimal solutions, which is often exponential. One way to reduce the computational time is to generate only a subset of the Pareto-optimal solutions, for instance with a K-best solutions algorithm [36]. The downside is that the set of solutions may not contain the most adequate Pareto-optimal solution with respect to the decision maker's preferences. In the case of the 1-HUC^{DRM} problem, the set of Pareto-optimal solutions for the BOSPP may not contain any solution of the 1-HUC^{DRM} problem. Hence, this algorithm can not be considered as such to provide the optimal solution, but we show in the following sections how we adapt the enumeration to our purpose.

Another way to generate fewer Pareto-optimal solutions is to consider a non-linear combination of the objectives, such as the Chebychev norm [86], a Choquet or Sugeno integrals [49], or Lorenz dominances [21]. The interest of considering such non-linear metrics is that they can lead to Pareto-optimal solutions without being limited to the Pareto-supported ones, in opposition to a linear metric. The downside is that one may need some decision maker's preferences to efficiently set up these metrics Also, introducing a non-linear function often induces large computational times. For our purpose, this type of algorithm seems to be impractical. Indeed, the optimal solution of the 1-HUC^{DRM} problem may not be a Pareto-optimal solution of the BOSPP, especially with tight resource window constraints defined by (3.2.6) and (3.2.7). Hence, even with a non-linear metric, one requires to enumerate the solutions, which means solving multiple times a non-linear problem with large computational times. Besides, instances of the 1-HUC^{DRM} problem can greatly vary from one to another, making it more difficult to set up the non-linear metrics for any instance.

Note however that for some nonlinear metrics, some efficient algorithms exist. This is the case when considering the most robust solution, i.e., optimizing the worst case scenario (or worst objective in the case of multi-objective optimization). Indeed, a pseudo-polynomial algorithm exists for the shortest path problems with two scenarios [108]. This algorithm is pseudo-polynomial as it is linear with respect to the value of the objective function. For the 1-HUC^{DRM} problem, the values can be very large, therefore this type of algorithm may be ineffective.

6.4 An exact A* variant for the discretized 1-Hydro Unit Commitment problem

In this section, we describe the new algorithm proposed to solve the 1-HUC^{DRM} problem. The aim of this algorithm is to find the longest path in graph G_E^* as described in Section 6.2.1. A difficulty of this graph is the pseudo-polynomial number of vertices, which is why we resort to a variant of the A* algorithm [50] to dynamically build G_E. The A* algorithm is particularly efficient when the number of vertices is large as it involves a dual bound to guide the search, and to discard suboptimal partial solutions. We denote the proposed exact variant of the A* algorithm for the 1-HUC^{DRM} problem by HA*. This algorithm involves a dedicated dual bound for the 1-HUC^{DRM} problem.

6.4.1 Dual bound

In the case of the 1-HUC^{DRM} problem, a dual bound overestimates the value of the objective function because we are solving a maximization problem. Let p be a path from time period 1 to t representing already taken decisions. The aim is to compute a dual bound from time period $t+1$ to T. The idea of the proposed dual bound is to compute an improved linear relaxation of model $M_{op\text{-}DRM}$ defined in **Section 3.3** on time periods $t+1$ to T. To do so, we define quadruplets $(t, i, val, flow)$, with t a time period, i an operating point, val the value $\sum_{j=0}^{i} \Psi_{t,j}$, and $flow$ the value $\sum_{j=0}^{i} D_j$. The aim is to progressively increase the values of variables $x_{t,i}$ depending on their profitability, being $val/flow$.

Algorithm 4 describes how to compute a linear relaxation on time periods $t+1$ to T from a partial path in graph G_E^*, as detailed in the following four steps.

Step 1: Initialize a fractional solution with $x_{t',i} = 1$ if the partial solution represented by path p requires operating point $i \leq N$ at time period $t' \leq t$, $x_{t',i} = 0$ otherwise. This step is represented by lines 1 to 8 of **Algorithm 4**.

Step 2: Initialize a list with all the quadruplets at time period $t' \in [t+1; T]$ by decreasing profitability $val/flow$. This step is represented by lines 9 to 14 of **Algorithm 4**.

Feasibility step 3: This step is repeated as long as lower bounding constraints (3.2.6) are not verified (the while loop at line 15 of **Algorithm 4**). The algorithm looks for the smallest time period t' such that (3.2.6) is not satisfied, and for $x_{t'',i}$ with $t'' \in [t+1; t']$ associated to $(t'', i, val, flow)$ maximizing profitability $val/flow$. The variable $x_{t'',i}$ is increased depending on its profitability:

- If the profitability is positive, fractionally increase $x_{t'',i}$ as much as possible provided all upper bounds α^* are satisfied.
- Otherwise, fractionally increase $x_{t'',i}$ as little as possible provided all upper bounds and the lower bounds $\beta^*_{t'}$ are satisfied.

All variables $x_{t'',i'} < x_{t'',i}$ with $i' < i$ must be set to the same value as $x_{t'',i}$ in order to satisfy the order constraints. Also, for any $i' > i$, quadruplets $(t'', i', val', flow')$ are updated as $(t'', i', val' - val, flow' - flow)$. This is because as $x_{t'',i} > x_{t'',i'}$, one can increase $x_{t'',i'}$ without increasing $x_{t'',i}$ while still verifying

order constraints. For all variables that cannot be further increased due to the upper bounds (3.2.7), the associated quadruplets are removed from the list.

Optimality step 4: This step is repeated as long as there is a variable of positive profitability associated to a quadruplet in the list of quadruplets (again the while loop at line 15 of **Algorithm 4**). Select the variable associated to the first quadruplet of the list, and fractionally increase its value as much as possible provided all upper bounds are satisfied. Remove from the list all quadruplets associated to a variable that cannot be further increased due to the upper bound (3.2.7).

Property 38

*The fractional solution returned by **Algorithm 4** verifies order constraints.*

Proof : Consider two quadruplets $(t, i, val, flow)$ and $(t, i', val', flow')$, with t a time period considered by **Algorithm 4** and $i' > i$. At the start of the algorithm, $x_{t,i} = x_{t,i'} = 0$. If $x_{t,i'}$ is increased, then $x_{t,i}$ is increased by the same amount (see line 22 of **Algorithm 4**), hence order constraints are satisfied. If $x_{t,i}$ is increased, it means that $val/flow > val'/flow'$. Hence, $val/flow > (val'-val)/(flow'-flow)$. Consequently, the algorithm will increase $x_{t,i'}$ only if $x_{t,i} = 1$, hence order constraints are satisfied. ■

Theorem 13

***Algorithm 4** defines a dual bound for the 1-HUC^{DRM} problem.*

Proof : Let s be an integer solution for the 1-HUC^{DRM} problem, for which variables $x_{t,i}$ satisfy all constraints of $M_{op\text{-}DRM}$. Let $\hat{s}$ be a fractional solution for the 1-HUC^{DRM} problem obtained with **Algorithm 4**, for which $\hat{x}_{t,i}$ verify all constraints of $M_{op\text{-}DRM}$ for time periods 1 to $t \leq T$. Consider $\hat{s}$ and s to be identical for time periods 1 to t.

Let $\mathcal{X}$ (resp. $\mathcal{Y}$) be the variables such that $x_{t',i} < \hat{x}_{t',i}\ \forall x_{t',i} \in \mathcal{X}$ (resp. $x_{t',i} > \hat{x}_{t',i}\ \forall x_{t',i} \in \mathcal{Y}$) with $t < t' \leq T$.

Clearly, for each variable $x_{t',i} \in \mathcal{X}$ of positive profitability, a fractional value for $x_{t',i}$ increases the value of the objective function compared to $x_{t',i} = 0$. Similarly, for each variable in $\mathcal{Y}$ of negative profitability, a fractional value for $x_{t',i}$ increases the value of the objective function compared to $x_{t',i} = 1$.

Let $\mathcal{X}^- \subseteq \mathcal{X}$ and $\mathcal{Y}^- \subseteq \mathcal{Y}$ be the variables with negative profitability. Suppose $|\mathcal{X}^-| > 0$. By construction of $\hat{s}$, the variables of $\mathcal{X}^-$ have value greater than 0 only in order to yield a feasible solution, with respect to a lower bound $\beta^*_{t'}$. In such a case the total flow of $\hat{s}$ from time period 1 to t' is exactly $\beta^*_{t'}$ by construction of $\hat{s}$. As solution s also verifies all lower bounds, we deduce $|\mathcal{Y}^-| > 0$. Otherwise, there is either a contradiction with the construction of $\hat{s}$, or s does not verify all lower bounds. Hence, the total flow of s from time period 1 to t' is at least $\beta^*_{t'}$. By definition, s and $\hat{s}$ are identical from time period 1 to t, consequently the only difference is on time periods $t+1$ to t'. By construction of $\hat{s}$ the variables of $\mathcal{X}^-$ are the most profitable and have fractional value in $\hat{s}$. Consequently, their weighted value in the objective function must be higher than those of $\mathcal{Y}^-$ in the integer solution s.

A similar proof can be made for variables in $\mathcal{Y}^+ \subseteq \mathcal{Y}$ and $\mathcal{X}^+ \subseteq \mathcal{X}$ the variables with positive profitability.

The value of $\hat{s}$ is then greater or equal to the value of s. ■

It is possible to tighten the fractional solution returned by **Algorithm 4** while keeping its value as a dual bound. Clearly, an integer solution is necessarily of a total flow which is a combination of the flows from the operating points. Therefore, the flow of an integer solution is necessarily a multiple of the Greatest Common Divisor (GCD) of the operating points' flows. When the algorithm increases the value of a variable, we can increase or reduce this value so that the total flow of the returned solution remains a multiple of the GCD relative to the operating points' flows. This is achieved at steps 18 and 20 of **Algorithm 4**. Note that since the flows are identical from one time period to another, we can quickly compute the GCD by considering only the flows of a single time period.

Algorithm 4 DualBound Algorithm

Require: A path p from time period 1 to t, a graph G_E^*, GCD the GCD of the flow
1: Initialize solution $\hat{x}$ with all variables to 0
2: **for** $t' \in [1;T]$ and $i \in [1;N]$ **do**
3: **if** $t' \leq t$ AND p requires operating point i at time period t' **then**
4: $\hat{x}_{t',i} = 1$
5: **else**
6: $\hat{x}_{t',i} = 0$
7: **end if**
8: **end for**
9: Initialize a list $L = []$
10: **for** $t' \in [t+1;T]$ and $i \in [1;N]$ **do**
11: $flow \leftarrow \sum_{j=1}^{i} D_j$
12: $val \leftarrow \sum_{j=1}^{i} \Psi_{t',j}$
13: add $(t', i, val, flow)$ in L, sorted by decreasing $val/flow$
14: **end for**
15: **while** $\exists t' \in [t+1, T]$ such that $\hat{x}$ does not verify $\beta_{t'}^*$ OR exists quadruplet in L with $val > 0$ **do**
16: $(t'', i, val, flow) \leftarrow$ first in L with $t'' \leq t'$
17: **if** $val \leq 0$ **then**
18: set $\hat{x}_{t'',i}$ to minimum such that $\beta_{t'}^*$ verified and $x_{t'',i} \cdot flow \mod GCD = 0$; if $\beta_{t'}^*$ cannot be verified $\hat{x}_{t'',i} \leftarrow 1$
19: **else**
20: set $\hat{x}_{t'',i}$ to maximum such that all upper bounds are verified and $x_{t'',i} \cdot flow \mod GCD = 0$
21: **end if**
22: **for** $i' \in [1;i]$ **do**
23: $\hat{x}_{t'',i'} = \max(\hat{x}_{t'',i'}, \hat{x}_{t'',i})$
24: **end for**
25: **for** $(t'', i', val', flow') \in L$ with $i' \in]i;N]$ **do**
26: $val' \leftarrow val' - val$
27: $flow' \leftarrow flow' - flow$
28: **end for**
29: **for** $(t'', i, val, flow) \in L$ such that $\hat{x}_{t'',i}$ cannot be increased **do**
30: remove $(t'', i, val, flow)$ from L
31: **end for**
32: **end whilereturn** the value of $\hat{x}$

6.4.2 Hydro A* algorithm

For a 1-HUC^{DRM} problem with an objective function to maximize, the principle of HA* is the following. Consider a pool of partial solutions evaluated with the dual bound. At each iteration, the partial solution with the highest dual bound value is considered and removed from the pool. From the partial solution considered, we complement it by adding neighbors relative to its last vertex. Once a solution is found, its value is used as a bound to remove some more partial solutions from the pool. Indeed, if the solution's value is higher than a partial solution's dual bound, then the partial solution can be removed from the pool. Once the pool of partial solutions is empty, the algorithm stops and the best solution found is the optimal solution.

We underline the need of a tight dual bound. If the bound used is not a dual bound, then one might discard feasible solutions, including the optimal one. If the dual bound is too loose, there are fewer cases where one can prune partial solutions while guaranteeing optimality. Hence, more vertices are developed which can exponentially increase the computational time.

For readability purposes, we introduce three structures:

Definition 51 (*Path structure*)

> *The path structure has three attributes: $vertices$ the list of vertices of the path; val the value of the path with respect to the objective function; $dual$ the dual bound value.*

Definition 52 (*Vertex structure*)

> *A vertex structure has two attributes: t the time period and d the cumulated flow $\mathcal{D}_{1,t}$, as defined for the vertices of G_E^* in Section 6.2.1.*

Definition 53 (*Arc structure*)

> *The arc structure has one attribute: val the value as defined for the arcs of G_E^* in Section 6.2.1.*

Algorithm 5 presents the pseudocode of HA*, using the three previously described structures as well as $DualBound$. The dominance rules used in **Algorithm 5** are the dominance rules 1 and 2 defined in **Section 6.2.1**.

In the following, we present experimental results to demonstrate the efficiency of the HA* algorithm against two state-of-the-art approaches.

6.4.3 Numerical results for the discretized 1-Hydro Unit Commitment without ramping nor min-up/down constraints

Results presented in **Section 6.5.3** show that HA* is not particularly efficient on generic instances of the 1-HUC^{DRM} problem. This is due to the algorithm computing the dual bound, which does not take into account ramping and min-up/down constraints (3.1.9)-(3.1.12). However, we identify that HA* becomes

Algorithm 5 Algorithm HA*

Require: A graph G_E^*
  Initialize a path p as follows: $p.vertices = \{(0,0)\}$, $p.val = 0.0$, $p.dual = DualBound(p)$
  Initialize a list of paths $L_p = [p]$
  Initialize the best solution $bestSol = \emptyset$
  Initialize the value of the best solution $bestVal = -\infty$
  while L_p not empty **do**
    $p \leftarrow$ first path in L_p
    remove p from L_p
    $v \leftarrow$ last vertex of $p.vertices$
    for arc a from v to u **do**
      $q.vertices = p.vertices \cup u$, $q.val = p.val + a.val$, $q.dual = DualBound(q)$
      if $|q.vertices| = T + 1$ **then**
        if $bestVal < q.val$ **then**
          $bestVal \leftarrow q.val$
          $bestSol \leftarrow q.vertices$
          remove $q' \in L_p$ with $q'.val + q'.dual \leq bestVal$
        end if
      else
        $dom \leftarrow FALSE$
        for $q' \in L_p$ **do**
          if q' dominates q **then**
            $dom \leftarrow TRUE$
          end if
          if q dominates q' **then**
            remove q' from L_p
          end if
        end for
        if $dom = FALSE$ and $q.val + q.dual > bestVal$ **then**
          add q in L_p by keeping L_p sorted by decreasing $q.val + q.dual$
        end if
      end if
    end for
  end while
  return $bestSol$

efficient for the special case of the 1-HUC^{D} problem. In the following, we show results for instances of the 1-HUC^{D} problem.

Following results are computed on a single thread of an Intel(R) Core(TM) i7-9850H CPU @ 2.60GHz processor, with 2 CPUs of 8 cores, with Linux as operating system. All algorithms are developed with C++. Version 12.8 of CPLEX with default setting is used to solve the MILP formulation.

From a large set of realistic instances derived from a real EDF plant, a first set A of 13 instances is obtained. These 13 instances are retained as preliminary results have shown that formulation $M_{op\text{-}D}$ is not trivially solved. This emphasizes the need of an efficient alternative in these cases. For each of these instances, $T = 96$, $L = 1$ and the ramping R_u, R_d are not restrictive. **Table 6.2** depicts for each instance the main characteristics, the number of operating points, the presence of a constraining minimum (resp. maximum) bound on the volume at the last time period and the presence of a tight resource window at the last time period. The instances cover three cases, namely when there is only an upper bound, only a lower bound, or a target volume with a constraining upper and lower bound. In the case of an equality constraint, the instance becomes infeasible, which is out of the scope of the instances considered in this chapter. Hence, the target volumes are not equality constraints, but rather tight window constraint. For instances with a target volume, more resource windows are obtained by propagating the bounds β^* and α^* from the bounds at the last time period to the previous ones.

The water flow D and power P are in the order of 10^3 to 10^4, with volumes in the order of 10^7. For target volumes, the difference between the upper and lower bound is in the order of 10^3, which is small enough with respect to the flows to yield very few vertices at time period T in G_E^*.

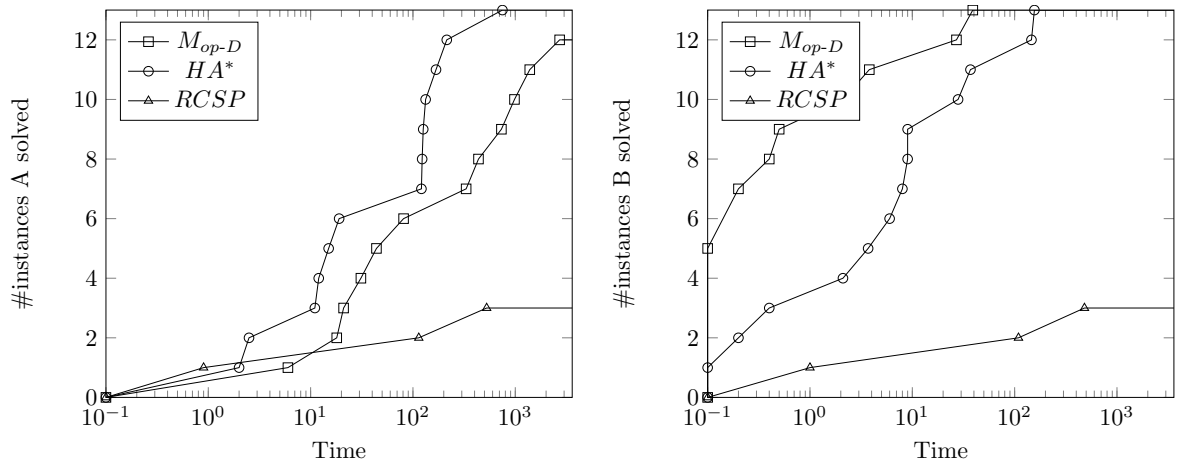
Table 6.2: Main characteristics of instances set A with $T = 96$ time periods

instance	N	minimum volume	maximum volume	tight window resources
1	4	✗	✓	✗
2	4	✓	✗	✗
3	4	✓	✓	✓
4	7	✗	✓	✗
5	7	✓	✗	✗
6	7	✓	✓	✓
7	8	✗	✓	✗
8	8	✓	✓	✓
9	15	✗	✓	✗
10	17	✗	✓	✗
11	18	✓	✗	✗
12	21	✗	✓	✗
13	18	✗	✓	✗

A second set B of 13 instances, similar to the first set, is also constructed. The only differences are bounds β_T^* and α_T^* that are shifted as follows. Let an instance with bounds $\beta_T^*[A]$ and $\alpha_T^*[A]$ be in set A. A random value $k \in [-9999;-1000] \cup [1000;9999]$ is chosen. Bounds of an instance for set B are $\beta_T^*[B] = \beta_T^*[A] + k$ and $\alpha_T^*[B] = \alpha_T^*[A] + k$. Note that this shift is very small for these instances. Indeed, the water flows can be in the order of 10^4, there are nearly 100 time periods and the cumulated flow $\mathcal{D}_{1,T}$ is in the order of 10^6. The shift k is at most 1% of $\overline{\mathcal{D}}_{1,T}$. As shown in the following results, slightly

modifying an instance can drastically impact the computational time.

In order to benchmark HA*, all instances are solved with HA* as well as with two alternative methods. The first alternative is a classical RCSPP algorithm [9] adapted to the 1-HUC^{D} problem [2]. The second alternative is to use CPLEX to solve $M_{op\text{-}D}$ described in **Section 3.3**. All algorithms use a single thread, with a time limit of one hour.



(a) Total number of instances A solved per computational time

(b) Total number of instances B solved per computational time

Figure 6.3: Total number of instances A and B solved per computational time

Figure 6.3a and **6.3b** represent the number of instances solved by each algorithm with respect to the computational time. Clearly, for instance set A, HA* is the most efficient alternative. Indeed, it solves every instance and requires less time than the other alternatives. For instance set B, solving $M_{op\text{-}D}$ is the most efficient alternative. Note that the difference between the solving times of $M_{op\text{-}D}$ and HA* is only of a few seconds for instance set B, as most instances are solved in less than 10 seconds with HA*. Solving $M_{op\text{-}D}$ is the least robust alternative when it comes to computational times. Indeed, when comparing the results between instance sets A and B, the computational times are drastically different for $M_{op\text{-}D}$, whereas for HA* there is a smaller difference, and for the RCSPP algorithm the results are the same. The RCSPP algorithm fails to solve 10 out of 13 instances, for both instance sets A and B. This shows that the RCSPP algorithm is inefficient at solving the 1-HUC^{D} problem. We further explain the results in the following, by introducing **Table 6.3** and **6.4** with detailed results.

Table 6.3 and **6.4** give, for each instance, the value of the objective function and the computational times obtained for all algorithms, as well as the optimality gap and the number of Branch & Bound nodes returned by the MILP solver. If the MILP solver proves optimality, the gap is noted "opt". When the time limit is reached, the time is noted "-". Results are presented for each instance individually, as well as the average (avg) and the standard deviation (sd) for the solved instances. When the time limit is reached, a computational time of 3600 seconds is accounted for in the average and the standard deviation. The computational time of the most efficient algorithm is emphasized in bold for each instance, as well as

the average and the standard deviation for each set of instances.

Table 6.3: Performance of solving $M_{op\text{-}D}$, the RCSPP algorithm and HA* on instance set A

	$M_{op\text{-}D}$				RCSPP		HA*	
instance	value	#nodes	gap	time	value	time	value	time
1	−25 428.80	10 232 857	opt	2713.4	-	-	−25 428.80	**2.5**
2	43 010.90	2 591 995	opt	437.0	43 010.9	**0.9**	43 010.90	2.3
3	3556.54	5 034 351	opt	1384.8	-	-	3556.54	**12.0**
4	2462.90	167 490	opt	44.1	-	-	2462.90	**11.1**
5	111 115.00	101 570	opt	**17.8**	111 115.0	525.2	111 115.00	18.8
6	−1706.62	888 735	opt	331.4	-	-	−1706.62	**14.9**
7	5692.65	115 626	opt	**6.2**	-	-	5692.65	120.6
8	16 581.10	487 218	opt	**30.9**	-	-	16 581.10	126.0
9	−71 645.90	176 123	opt	**21.4**	-	-	−71 645.90	133.0
10	−525 446.00	901 533	opt	732.1	-	-	−525 446.00	**168.4**
11	44 421.20	1 535 139	opt	982.0	44 421.2	**114.1**	44 421.20	213.9
12	−20 329.90	1 624 614	0.58	-	-	-	−20 324.00	**748.8**
13	−103 435.00	365 253	opt	**80.7**	-	-	−103 435.00	122.8
avg	-	1 865 577	0.58	798.6	-	1431.0	-	**130.4**
sd	-	2 755 062	0.0	1101.1	-	2818.4	-	**191.7**

Table 6.4: Performance of solving $M_{op\text{-}D}$, the RCSPP algorithm and HA* on instance set B

	$M_{op\text{-}D}$				RCSPP		HA*	
instance	value	#nodes	gap	time	value	time	value	time
1	−25 430.30	873	opt	**0.2**	-	-	−25 430.30	0.4
2	42 993.00	0	opt	**0.0**	42 993.00	1.0	42 993.00	**0.0**
3	3700.23	11	opt	**0.0**	-	-	3700.23	0.2
4	2462.90	16 330	opt	**2.1**	-	-	2462.90	3.7
5	111 143.00	34 982	opt	**3.8**	111 143.00	481.5	111 143.00	8.4
6	−1460.33	2519	opt	**0.4**	-	-	−1460.33	2.1
7	5936.31	522 284	opt	**39.2**	-	-	5936.31	155.7
8	16 730.40	6403	opt	**0.5**	-	-	16 730.40	8.6
9	−132 290.00	0	opt	**0.0**	-	-	−132 290.00	9.2
10	−525 246.00	46 425	opt	**27.1**	-	-	−525 246.00	37.2
11	44 646.80	0	opt	**0.0**	44 646.80	109.1	44 646.80	6.2
12	−20 153.10	1586	opt	**0.1**	-	-	−20 153.10	146.3
13	−103 339.00	2314	opt	**0.2**	-	-	−103 339.00	27.7
avg	-	48 748	-	**5.7**	-	1437.1	-	31.2
sd	-	137 446	-	**12.0**	-	2814.7	-	52.2

There is a clear difference in computational time between set A and B. Indeed, $M_{op\text{-}D}$ is solved for 12 out of the 13 instances of set A, and needs between 6 and 2713 seconds, while it is solved for all instances of set B, most of them instantaneously. Similarly, HA* solves all instances of set A and needs between 2 and 748 seconds, while it solves all instances of set B in less than 156 seconds. Note however

that there is no noticeable computational time difference between solving instance set A and B with the RCSPP algorithm.

The RCSPP algorithm is only able to solve instances 2, 5 and 11, for both sets. This is because for any other instance, the maximum volume of the upstream reservoir is constrained at the last time period. In this case, the value β_T^* becomes positive, meaning that there is a minimum bound on the resource. As mentioned in [3], when there is a minimum bound on the resource, the dominance properties of the RCSPP algorithm cannot always be applied, thus leading to large computational times. Clearly, HA* outperforms the RCSPP algorithm. Even when there is no minimum bound on the resource, the RCSPP algorithm yields larger computational times for all instances of set B and instance 5 of set A.

When comparing HA* algorithm to solving $M_{op\text{-}D}$ on the most difficult instances, the former outperforms the latter in terms of computational time and number of instances solved. On the one hand, HA* is more stable with respect to the computational times. Indeed, HA* only requires more than 10 minutes once (instance 12 of set A), while solving $M_{op\text{-}D}$ requires more than 10 minutes for 5 of the 26 instances (instances 1, 3, 10, 11 and 12 of set A). Moreover, $M_{op\text{-}D}$ is not solved to optimality for instance 12 of set A, and the best value found is not the optimal value obtained with HA*. On the other hand, solving $M_{op\text{-}D}$ appears to be more efficient on easier instances than HA*. In this case, there are numerous instances where the difference between the two approaches is within a few seconds (instances 1, 2, 3, 4 and 6 of set B).

The stability with respect to the computational time is noticeable on the average and standard deviation. Indeed, the average time difference between set A and B is much smaller for HA* than for the MILP solver. The standard deviation for HA* is much smaller on set A, and slightly larger for set B when compared to solving $M_{op\text{-}D}$. Besides, one can compute the total average and total deviation of the solution time for all 26 instances. The total average time and standard deviation are 402.1 seconds and 873.7 seconds for the MILP solver, and 80.3 seconds and 149.0 seconds for HA*.

6.5 A two-phase method for the discretized 1-Hydro Unit Commitment problem

Even though algorithm HA* is efficient for the 1-HUC^{D} problem, it is not the case for the 1-HUC^{DRM} problem. First, the graph is larger with more vertices to represent these constraints. Second, ramping and min-up/down constraints make the operating points, that can be reached at a time period t, depend on the partial path from time period 1 to $t-1$, meaning that the dominance rules apply in fewer cases. Third, the dual bound does not take into account min-up/down and ramping constraints. Hence, its value becomes too loose which negatively impacts the performances of HA*.

In this section, we present a new graph algorithm that can handle the 1-HUC^{DRM} problem, and more generally can solve an RWSPP with a single resource. To do so, we first define the bi-objective relaxation of the RWSPP. Then, we describe the core ideas, inspired by the two-phase algorithm for bi-objective problems [103] to solve the RWSPP. The idea is to enumerate solutions of the bi-objective relaxation

of the RWSPP in a reduced search space. Finally, this algorithm is benchmarked on real 1-HUC^{DRM} problem instances, against three state-of-the-art approaches.

6.5.1 Bi-objective relaxation of the Longest Path Problem with Resource Windows

In this section, we first define some properties for the solutions of a Bi-Objective Longest Path Problem (BOLPP), which is similar to the BOSPP defined in **Section 6.1** but where the aim is to maximize the two objectives. Then, we define a *bi-objective relaxation* of the RWLPP as a BOLPP. Clearly each solution of the RWLPP is also a solution of the BOLPP as there is no resource constraints in the latter. Besides, we show in the following that one can obtain an upper bound for a solution of the RWLPP by a combination of the objectives of the BOLPP.

Definition 54 (*Aggregated value*)

We define $\mu_\delta(C) = \delta_1 \cdot V_1(C) + \delta_2 \cdot V_2(C)$ as the aggregated value of a path C considering vector δ.

Note that for any Pareto-supported path C, there exists a vector δ such that $\mu_\delta(C) \geq \mu_\delta(C')$ for any other path C'.

Remark 9

Without loss of generality, we consider $\delta_1 > 0$ and $\delta_2 > 0$. Indeed, consider two paths C and C' such that $V_1(C) > V_1(C')$ and $V_2(C) = V_2(C')$. In the case $\delta_1 \leq 0$, then $\mu_\delta(C) \leq \mu_\delta(C')$, meaning that C' could be Pareto-supported, while solution C dominates C' as the BOLPP aims to maximize both objectives. A similar observation can be done for δ_2. Also, note that the sum $\delta_1 + \delta_2$ does not necessarily equals 1, meaning that the aggregated value is not necessarily a convex combination of the two objectives.

Remark 10

The objective functions are not normalized before the aggregation. This is because when instances of some problems, such as those relative to the 1-HUC^{DRM} problem, are subject to numerical issues, they are sensitive to data scaling.

In the following, we denote P the ordered set of Pareto-supported paths, ordered by increasing value of the first objective. We do not consider in P two solutions with the exact same values, meaning that for two paths C and C' in P, $C \neq_p C'$.

We define the bi-objective relaxation of the RWLPP as the BOLPP with $V_1(C) = V(C)$ and $V_2(C) = R(C)$. The idea is to define a restricted search space to enumerate the feasible paths of the BOLPP, until the optimal path of the RWLPP is obtained. In the original two-phase algorithm for bi-objective problems, the search spaces are defined as triangle between consecutive Pareto-supported paths in P. We show later in this section an example where no optimal solution of the RWLPP is present in any of these triangles. However, we still can use Pareto-supported solutions and the resource window $[\underline{R}(p); \overline{R}(p)]$ to define a region.

First, we can introduce an upper bound for $V_1(C^*)$ with C^* the optimal path of the RWLPP.

Property 39

Let C be the Pareto-supported solution maximizing $V_1(C)$ and C^* the optimal path of the RWLPP. Then, $V_1(C)$ is a valid upper bound for $V_1(C^*)$

Proof : As C^* is a valid solution for the RWLPP, it is also a valid solution for the BOLPP. Consequently, if $V_1(C^*) > V_1(C)$, then C cannot be the Pareto-supported solution maximizing $V_1(C)$. ■

However, it can be possible to provide a tighter upper bound. We first introduce the following lemma.

Lemma 4

Let C_1, C_2 and C_3 be three solutions of the BOLPP with $V_1(C_1) \leq V_1(C_2) \leq V_1(C_3)$. Consider that these solutions do not dominate each other. Let δ be such that $\mu_\delta(C_1) = \mu_\delta(C_3)$. If $\mu_\delta(C_2) < \mu_\delta(C_1)$ then C_2 cannot be Pareto-supported.

Proof : Consider a vector δ' such that $\delta'_1 > \delta_1$ and $\delta'_2 = \delta_2$. Then, as $\mu_\delta(C_2) < \mu_\delta(C_3)$ and $V_1(C_2) \leq V_1(C_3)$, hence $\mu_{\delta'}(C_2) \leq \mu_{\delta'}(C_3)$. Consider a vector δ' such that $\delta'_1 < \delta_1$ and $\delta'_2 = \delta_2$. Then, as $\mu_\delta(C_2) < \mu_\delta(C_1)$ and $V_1(C_1) \leq V_1(C_2)$, hence $\mu_{\delta'}(C_2) \leq \mu_{\delta'}(C_1)$.

From a vector δ', one can obtain any other vector δ'' by multiplying δ'_1 and δ'_2 by the same non-negative real number. Hence, there is no vector δ'' for which C_2 maximizes $\mu_{\delta''}(C_2)$, hence C_2 is not Pareto-supported by definition. ■

We introduce **Properties 40** to **42** and **Theorem 14** to provide a tighter upper bound for $V_1(C^*)$.

Property 40

Let (C_1, C_2) be a pair of consecutive paths in P. Let δ be such that $\mu_\delta(C_1) = \mu_\delta(C_2)$. For each feasible path C_3 of the BOLPP, $V_1(C_3) \leq \frac{\mu_\delta(C_1) - \delta_2 \cdot V_2(C_3)}{\delta_1}$

Proof : Suppose that $V_1(C_3) > \frac{\mu_\delta(C_1) - \delta_2 \cdot V_2(C_3)}{\delta_1}$. Note that in such a case, $\mu_\delta(C_3) > \mu_\delta(C_1)$.

Consider the case $V_1(C_1) < V_1(C_3) < V_1(C_2)$ illustrated by **Figure 6.4a**. In such a case, C_1 and C_2 cannot be two consecutive paths in P by definition. Indeed, either C_3, or a Pareto-supported solution dominating C_3 is between C_1 and C_2 in P.

Consider the case $V_1(C_1) < V_1(C_2) < V_1(C_3)$ illustrated by **Figure 6.4b**. Let δ' be such that $\mu_{\delta'}(C_1) = \mu_{\delta'}(C_3)$. We suppose that $\mu_{\delta'}(C_2) \geq \mu_{\delta'}(C_3)$ to show a contradiction. We then know the following:

$$\delta_1 \cdot V_1(C_1) + \delta_2 \cdot V_2(C_1) = \delta_1 \cdot V_1(C_2) + \delta_2 \cdot V_2(C_2)$$
$$\delta_1 \cdot V_1(C_1) + \delta_2 \cdot V_2(C_1) < \delta_1 \cdot V_1(C_3) + \delta_2 \cdot V_2(C_3)$$
$$\delta'_1 \cdot V_1(C_1) + \delta'_2 \cdot V_2(C_1) = \delta'_1 \cdot V_1(C_3) + \delta'_2 \cdot V_2(C_3)$$
$$\delta'_1 \cdot V_1(C_1) + \delta'_2 \cdot V_2(C_1) \leq \delta'_1 \cdot V_1(C_2) + \delta'_2 \cdot V_2(C_2)$$

From the equations, we can deduce $\delta_1 = \delta_2 \cdot \frac{V_2(C_1) - V_2(C_2)}{V_1(C_2) - V_1(C_1)}$ and $\delta'_1 = \delta'_2 \cdot \frac{V_2(C_1) - V_2(C_3)}{V_1(C_3) - V_1(C_1)}$. From the inequalities, we can deduce $\delta_2 \cdot (V_2(C_1) - V_2(C_3)) < \delta_1 \cdot (V_1(C_3) - V_1(C_1))$ and $\delta'_2 \cdot (V_2(C_1) - V_2(C_2)) \leq \delta'_1 \cdot (V_1(C_2) - V_1(C_1))$

Combining these results, we obtain on one hand:

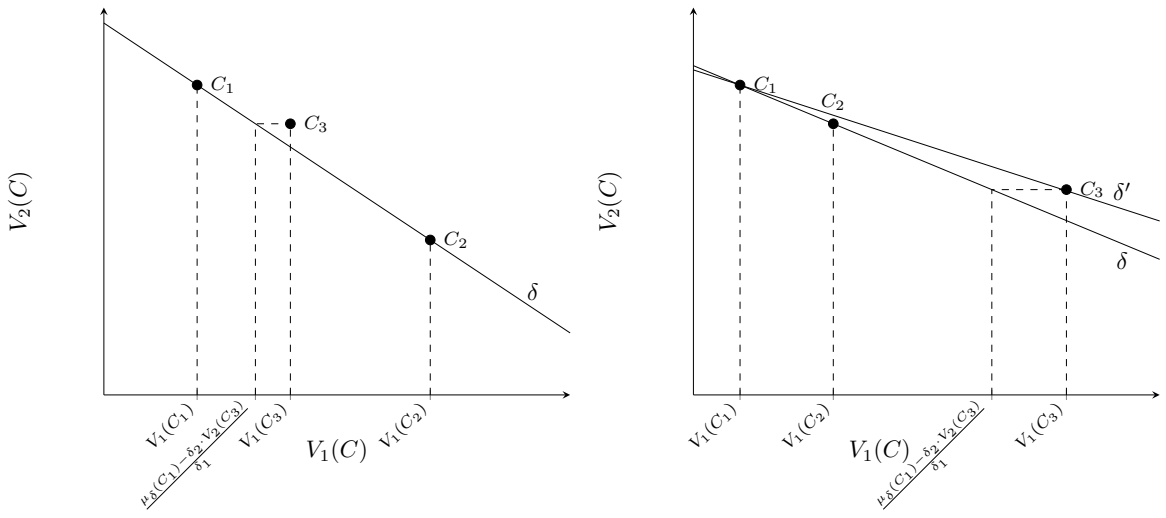
$$\delta_2 \cdot (V_2(C_1) - V_2(C_3)) < \delta_2 \cdot \frac{V_2(C_1) - V_2(C_2)}{V_1(C_2) - V_1(C_1)} \cdot (V_1(C_3) - V_1(C_1))$$
$$\frac{V_2(C_1) - V_2(C_3)}{V_2(C_1) - V_2(C_2)} < \frac{V_1(C_3) - V_1(C_1)}{V_1(C_2) - V_1(C_1)}$$

and on the other hand:

$$\delta'_2 \cdot (V_2(C_1) - V_2(C_2)) \leq \delta'_2 \cdot \frac{V_2(C_1) - V_2(C_3)}{V_1(C_3) - V_1(C_1)} \cdot (V_1(C_2) - V_1(C_1))$$
$$\frac{V_2(C_1) - V_2(C_2)}{V_2(C_1) - V_2(C_3)} \leq \frac{V_1(C_2) - V_1(C_1)}{V_1(C_3) - V_1(C_1)}$$
$$\frac{V_2(C_1) - V_2(C_3)}{V_2(C_1) - V_2(C_2)} \geq \frac{V_1(C_3) - V_1(C_1)}{V_1(C_2) - V_1(C_1)}$$

which contradict each-other. Hence $V_1(C_1) < V_1(C_2) < V_1(C_3)$ and implies that $\mu_{\delta'}(C_2) < \mu_{\delta'}(C_3)$. **Lemma 4** shows that in such a case C_2 cannot be Pareto-supported.

A similar proof can be done in the case $V_1(C_3) < V_1(C_1) < V_1(C_2)$, where C_1 cannot be Pareto-supported. ■



(a) Case $V_1(C_1) < V_1(C_3) < V_1(C_2)$ (b) Case $V_1(C_1) < V_1(C_2) < V_1(C_3)$

Figure 6.4: Illustrations for **Property 40**

Corollary 2

Let (C_1, C_2) be a pair of paths in P. Let δ be a vector such that $\mu_\delta(C_1) = \mu_\delta(C_2)$. Let C_3 be the solution maximizing $\mu_\delta(C_3)$. If path $C_3 \neq_p C_1$ and $C_3 \neq_p C_2$, then $V_1(C_3) \in]V_1(C_1); V_1(C_2)[$ and $V_2(C_3) \in]V_2(C_2); V_2(C_1)[$. If path $C_3 =_p C_1$ or $C_3 =_p C_2$, then C_1 and C_2 are consecutive paths in P.

Proof : The proof of **Property 40** indicates that if $C_3 \neq_p C_1$ and $C_3 \neq_p C_2$, then $V_1(C_3) \in]V_1(C_1); V_1(C_2)[$ and $V_2(C_3) \in]V_2(C_2); V_2(C_1)[$. Otherwise, either C_1 or C_2 cannot be Pareto-supported, which yields a contradiction.

In the case $C_3 =_p C_1$ or $C_3 =_p C_2$, then clearly there cannot be any Pareto-supported path C_4 with $V_1(C_4) \in]V_1(C_1); V_1(C_2)[$ as it cannot be part of the solutions' convex hull in the objective space. ■

For a solution C of the BOLPP, provided we know the value $V_2(C)$, this property gives an upper bound for $V_1(C)$ for each pair of consecutive paths in P. As the optimal path C^* of the RWLPP is also a feasible solution of the BOLPP, it also applies to C^*. As we do not know the value $V_2(C^*)$, we cannot use this property directly. However, we know that $V_2(C^*) \in [\underline{R}(p); \overline{R}(p)]$.

Property 41

Let (C_1, C_2) be a pair of consecutive paths in P. Let δ be such that $\mu_\delta(C_1) = \mu_\delta(C_2)$. Let $\overline{V}_1(C_1, \delta) = \frac{\mu_\delta(C_1) - \delta_2 \cdot \underline{R}(p)}{\delta_1}$ be the value on the first objective of the point intersecting the line going through C_1 with coefficient δ and the bound $\underline{R}(p)$. Then $\overline{V}_1(C_1, \delta)$ is an upper bound for the value of the optimal path $V_1(C^*)$ of the RWLPP.

Proof : As C^* is a feasible solution of the RWLPP, it is also a feasible solution of the BOLPP. From **Property 40**, we know that $V_1(C^*) \leq \frac{\mu_\delta(C_1) - \delta_2 \cdot V_2(C^*)}{\delta_1}$. As C^* is a feasible solution of the RWLPP, then $V_2(C^*) \in [\underline{R}(p); \overline{R}(p)]$, meaning that $V_2(C^*) \geq \underline{R}(p)$. Hence, we can deduce $V_1(C^*) \leq \frac{\mu_\delta(C_1) - \delta_2 \cdot V_2(C^*)}{\delta_1} \leq \frac{\mu_\delta(C_1) - \delta_2 \cdot \underline{R}(p)}{\delta_1}$, i.e., $V_1(C^*) \leq \overline{V}_1(C_1, \delta)$. ■

Property 41 provides an upper bound that can be computed independently from the values $V_1(C^*)$ and $V_2(C^*)$. Besides, we can also prove that for any $R \neq \underline{R}(p)$, the bound would be either looser or invalid.

Property 42

Let (C_1, C_2) be a pair of consecutive paths in P. Let δ be such that $\mu_\delta(C_1) = \mu_\delta(C_2)$. For any $R \leq \underline{R}(p)$, $\overline{V}_1(C_1, \delta) \leq \frac{\mu_\delta(C_1) - \delta_2 \cdot R}{\delta_1}$. For any $R > \underline{R}(p)$, $\frac{\mu_\delta(C_1) - \delta_2 \cdot R}{\delta_1}$ is not a valid upper bound for the value of the optimal solution $V_1(C^*)$ of the RWLPP

Proof : Clearly, for any $R \leq \underline{R}(p)$, $\frac{\mu_\delta(C_1) - \delta_2 \cdot \underline{R}(p)}{\delta_1} \leq \frac{\mu_\delta(C_1) - \delta_2 \cdot R}{\delta_1}$. Consider the case $R > \underline{R}(p)$. As $V_2(C^*) \geq \underline{R}(p)$, there is no guarantee that $R \leq V_2(C^*)$. Hence, it is not possible to deduce $V_1(C^*) \leq \frac{\mu_\delta(C_1) - \delta_2 \cdot R}{\delta_1}$ as done in **Property 41**. The following **Example 32** gives a counter-example to complete this proof in the case $R > \underline{R}(p)$. ■

Example 32

Consider an instance of the 1-HUC problem with $N = 2$ operating points, $T = 1$ time period, $R_u = R_d = +\infty$. The bounds on the cumulated flow are $\beta_1^* = 1$, $\alpha_1^* = 2$. The flow and power of each operating point are as follows: $(D_0 = 0, P_0 = 0)$, $(D_1 = 1, P_1 = 1 - \epsilon)$ with ϵ a small positive number and $(D_2 = 3, P_2 = 3)$. Power and water values are $\Lambda_1 = 1$, $\Phi^1 = 2$ and $\Phi^2 = 0$, meaning that the value of the

operating points are $\Psi_{1,0} = 0$, $\Psi_{1,1} = -1-\epsilon$ and $\Psi_{1,2} = -3$.

In graph G_C there are three distinct paths corresponding to the three operating points since $T = 1$. We denote C_0 (resp. C_1, C_2) the path going through a vertex associated to operating point 0 (resp. 1, 2). The values of these solutions are $V_1(C_0) = V_2(C_0) = 0$, $V_1(C_1) = -1-\epsilon$, $V_2(C_1) = 1$ and $V_1(C_2) = -3$, $V_2(C_2) = 3$. Clearly, solutions C_0 and C_2 are Pareto-supported, but not C_1. Hence, the list of Pareto-supported paths P is $[C_2, C_0]$. We select vector $\delta = [1,1]$ such that $\mu_\delta(C_0) = \mu_\delta(C_2)$. Also, due to the bounds $\beta_1^* = 1$, $\alpha_1^* = 2$, only C_1 is a feasible solution of the RWLPP, hence is the optimal solution.

By construction, $\underline{R}(p) = \beta_1^* = 1$ and $\overline{R}(p) = \alpha_1^* = 2$. Let R be any number such that $R > \underline{R}(p)$. For any value R, $\frac{\mu_\delta(C_0)-\delta_2 \cdot R}{\delta_1} = -R$. For this instance, one can always select ϵ small enough such that $-1-\epsilon > -R$. Consequently, if $R > \underline{R}(p)$, one cannot deduce a valid upper bound for the value $V_1(C^*)$ of the optimal path of the RWLPP.

From the previous **Properties 41** and **42**, we know that for a given pair (C_1, C_2) of consecutive paths in P, there is no $R \neq \underline{R}(p)$ providing a tighter upper bound computed as $\overline{V}_1(C_1, \delta)$. Recall that the number of Pareto-supported solutions can be exponential, hence one needs to compute an exponential number of bounds. We prove in the following that it is also possible to identify the pair (C_1, C_2) of consecutive paths in P that provides the tightest bound $\overline{V}_1(C_1, \delta)$.

Theorem 14

Let (C_1, C_2) be a pair of consecutive paths in P such that $V_2(C_1) \geq \underline{R}(p) > V_2(C_2)$. Let δ be such that $\mu_\delta(C_1) = \mu_\delta(C_2)$. Path C_1 and vector δ minimize the value $\overline{V}_1(C_1, \delta)$.

Proof : Consider another pair (C_1', C_2') of consecutive paths in P, and δ' such that $\mu_{\delta'}(C_1') = \mu_{\delta'}(C_2')$. As (C_1, C_2) and (C_1', C_2') are pairs of consecutive paths in P, then either $V_1(C_1) < V_1(C_2) \leq V_1(C_1') < V_1(C_2')$ or $V_1(C_1') < V_1(C_2') \leq V_1(C_1) < V_1(C_2)$.

Consider the case $V_1(C_1) < V_1(C_2) \leq V_1(C_1') < V_1(C_2')$ illustrated by **Figure 6.5a**. Suppose $\overline{V}_1(C_1', \delta') < \overline{V}_1(C_1, \delta)$, meaning that $\frac{\mu_{\delta'}(C_1')-\delta_2' \cdot \underline{R}(p)}{\delta_1'} < \frac{\mu_\delta(C_1)-\delta_2 \cdot \underline{R}(p)}{\delta_1}$. We can obtain the following result:

$$\frac{\mu_{\delta'}(C_1') - \delta_2' \cdot \underline{R}(p)}{\delta_1'} < \frac{\mu_\delta(C_1) - \delta_2 \cdot \underline{R}(p)}{\delta_1}$$

$$\frac{\mu_{\delta'}(C_1')}{\delta_1'} - \frac{\delta_2'}{\delta_1'} \cdot \underline{R}(p) < \frac{\mu_\delta(C_1)}{\delta_1} - \frac{\delta_2}{\delta_1} \cdot \underline{R}(p)$$

$$\frac{\mu_{\delta'}(C_1')}{\delta_1'} - \frac{\mu_\delta(C_1)}{\delta_1} < (\frac{\delta_2'}{\delta_1'} - \frac{\delta_2}{\delta_1}) \cdot \underline{R}(p)$$

Also, using **Property 40**, we know that $V_1(C_1) \leq \frac{\mu_{\delta'}(C_1')-\delta_2' \cdot V_2(C_1)}{\delta_1'}$, otherwise C_1' or C_2' are not Pareto-supported. Recall that $\mu_\delta(C_1) = \delta_1 \cdot V_1(C_1) + \delta_2 \cdot V_2(C_1)$, meaning that $V_1(C_1) = \frac{\mu_\delta(C_1)-\delta_2 \cdot V_2(C_1)}{\delta_1}$ and:

$$\frac{\mu_\delta(C_1) - \delta_2 \cdot V_2(C_1)}{\delta_1} \leq \frac{\mu_{\delta'}(C_1') - \delta_2' \cdot V_2(C_1)}{\delta_1'}$$

$$\frac{\mu_\delta(C_1)}{\delta_1} - \frac{\delta_2}{\delta_1} \cdot V_2(C_1) \leq \frac{\mu_{\delta'}(C_1')}{\delta_1'} - \frac{\delta_2'}{\delta_1'} \cdot V_2(C_1)$$

$$\left(\frac{\delta_2'}{\delta_1'} - \frac{\delta_2}{\delta_1}\right) \cdot V_2(C_1) \leq \frac{\mu_{\delta'}(C_1')}{\delta_1'} - \frac{\mu_\delta(C_1)}{\delta_1}$$

Using **Property 40**, we know that $V_1(C_2) \leq \frac{\mu_{\delta'}(C_1') - \delta_2' \cdot V_2(C_2)}{\delta_1'}$. Recall that $\mu_\delta(C_2) = \mu_\delta(C_1) = \delta_1 \cdot V_1(C_2) + \delta_2 \cdot V_2(C_2)$, meaning that $V_1(C_2) = \frac{\mu_\delta(C_1) - \delta_2 \cdot V_2(C_2)}{\delta_1}$. We deduce in the similar fashion the following inequality:

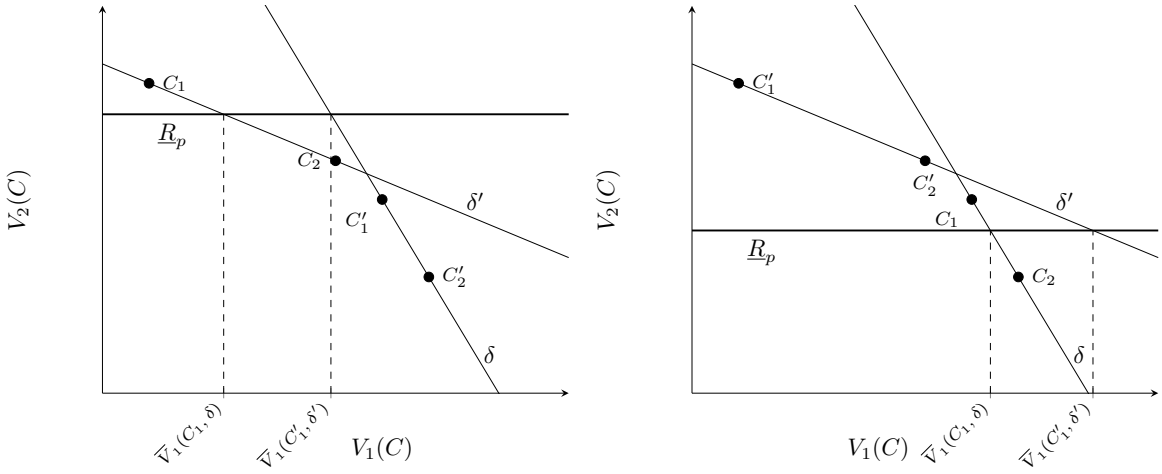
$$\left(\frac{\delta_2'}{\delta_1'} - \frac{\delta_2}{\delta_1}\right) \cdot V_2(C_2) \leq \frac{\mu_{\delta'}(C_1')}{\delta_1'} - \frac{\mu_\delta(C_1)}{\delta_1}$$

To summarize, we obtained the following inequalities:

$$\left(\frac{\delta_2'}{\delta_1'} - \frac{\delta_2}{\delta_1}\right) \cdot V_2(C_1) \leq \frac{\mu_{\delta'}(C_1')}{\delta_1'} - \frac{\mu_\delta(C_1)}{\delta_1} < (\frac{\delta_2'}{\delta_1'} - \frac{\delta_2}{\delta_1}) \cdot \underline{R}(p)$$
$$\left(\frac{\delta_2'}{\delta_1'} - \frac{\delta_2}{\delta_1}\right) \cdot V_2(C_2) \leq \frac{\mu_{\delta'}(C_1')}{\delta_1'} - \frac{\mu_\delta(C_1)}{\delta_1} < (\frac{\delta_2'}{\delta_1'} - \frac{\delta_2}{\delta_1}) \cdot \underline{R}(p)$$

As $V_2(C_2) < \underline{R}(p) \leq V_2(C_1)$, then there must be a contradiction. Consequently, $\overline{V}_1(C_1', \delta') \geq \overline{V}_1(C_1, \delta)$.

In a similar way, we can prove that $\overline{V}_1(C_1', \delta') \geq \overline{V}_1(C_1, \delta)$ in the case $V_1(C_1') < V_1(C_2') \leq V_1(C_1) < V_1(C_2)$ as shown in **Figure 6.5b**. ■



(a) Case $V_1(C_1) < V_1(C_2) \leq V_1(C_1') < V_1(C_2')$ (b) Case $V_1(C_1') < V_1(C_2') \leq V_1(C_1) < V_1(C_2)$

Figure 6.5: Illustrations for **Theorem 14**

Let (C_1, C_2) be a pair of consecutive paths in P such that $V_2(C_1) \geq \underline{R}(p) > V_2(C_2)$. Let C be the path maximizing $V_1(C)$. Then $\overline{V}_1(C_1, \delta) < V_1(C)$. Indeed, as $\underline{R}(p) > V_2(C_2)$, then $\overline{V}_1(C_1, \delta) < V_1(C_2)$ and $V_1(C_2) \leq V_1(C)$.

Note that **Theorem 14** may not apply to all instances. Indeed, if the instance has a feasible solution, then there is necessarily a Pareto-supported path C with $V_2(C) > \underline{R}(p)$. However, there is not necessarily a Pareto-supported path C with $V_2(C) \leq \underline{R}(p)$. We can distinguish three cases of the RWLPP: Locally Infeasible by Excess (LIE), Locally Infeasible by Deficiency (LID) and Locally Feasible (LF).

Definition 55 (*LIE, LID and LF*)

Let C be the path maximizing $V(C)$ without taking into account the resource windows. Case LID is when $R(C) < \underline{R}(p)$, case LIE is when $R(C) > \overline{R}(p)$ and case LF is with $R(C) \in [\underline{R}(p); \overline{R}(p)]$.

Property 43

Consider an instance of the RWLPP in case LIE or LF. Consider its associated bi-objective relaxation, being a BOLPP with $V_1(C) = V(C)$ and $V_2(C) = R(C)$ for any path C. Let C be the path maximizing $V_1(C)$. Consider a pair of consecutive paths (C_1, C_2) of P, and δ such that $\mu_\delta(C_1) = \mu_\delta(C_2)$. The bound $\overline{V}_1(C_1, \delta) \geq V_1(C)$.

Proof : Suppose there is a pair (C_1, C_2) of consecutive paths of P such that $\overline{V}_1(C_1, \delta) < V_1(C)$. As we consider case LIE or LF, then $V_2(C) > \underline{R}(p)$. We deduce $\frac{\mu_\delta(C_1 - \delta_2 \cdot V_2(C_1))}{\delta_1} < \frac{\mu_\delta(C_1 - \delta_2 \cdot \underline{R}(p))}{\delta_1} < V_1(C)$, which contradicts **Property 40**. Hence, $\overline{V}_1(C_1, \delta) \geq V_1(C)$. ■

Let (C_1, C_2) be a pair of consecutive paths in P such that $V_2(C_1) \geq \underline{R}(p) > V_2(C_2)$. Let C be the path maximizing $V_1(C)$. In case LID, bound $\overline{V}_1(C_1, \delta)$ is the best bound, whereas in case LIE and LF, bound $V_1(C)$ is the best bound.

Note that all **Properties 39** to **43** also hold when the values are negative. This means that one can build an *inverted bi-objective relaxation* of the RWLPP, being the BOLPP with $V_1(C) = V(C)$ and $V_2(C) = -R(C)$. Results from **Properties 39** and **40** as well as **Corollary 2** hold for the inverted bi-objective relaxation. Results from **Property 41** can be adapted by considering $-\overline{R}(p)$ instead of $\underline{R}(p)$ in the definition of $\overline{V}_1(C_1, \delta)$. For the following, the upper bound in the inverted case is denoted $\overline{V}^I_1(C_1, \delta)$. Similarly, **Property 42** and **Theorem 14** can both be adapted by considering $-\overline{R}(p)$ instead of $\underline{R}(p)$. For the inverted bi-objective relaxation, the result of **Property 43** changes, as $\overline{V}^I_1(C_1, \delta) \geq V_1(C)$ is true in case LID or LF, rather than LIE or LF.

Hence, in case LID, the best upper bound is $\overline{V}_1(C_1, \delta)$ obtained from the bi-objective relaxation; in case LIE the best bound is $\overline{V}^I_1(C_1, \delta)$ obtained from the inverted bi-objective relaxation; and in case LF, it is $V_1(C)$ with C the Pareto-supported solution maximizing $V_1(C)$. One can induce a search space using these bounds and the fact that the optimal solution of the RWLPP is with $V_2(C^*) \in [\underline{R}(p); \overline{R}(p)]$.

Figure 6.6 shows three instances of the RWLPP on a same graph. An arc a is annotated by a single pair $(V(a), R(a))$ if the values are the same for the three instances, or by three pairs $(V(a), R(a))$ if the values change between the instances. Each vertex v is annotated by the resource window $[\underline{R}(v); \overline{R}(v)]$. **Table 6.5** enumerates each path and its value and resource for each of the three instances. Note that for each of these instances, the path maximizing the value is C_C. For all three instances, $\underline{R}(p) = 29$ and $\overline{R}(p) = 34$. Hence the first instance is in case LID as $R(C_C) = 22$, the second instance in case LF as $R(C_C) = 31$ and the third instance in case LIE as $R(C_C) = 35$. **Figures 6.7a** to **Figures 6.7c** show the solutions' space of the bi-objective relaxation, the triangles from the classical bi-objective two-phase algorithm, the bound $\overline{V}_1(C_1, \delta)$ as well as our search space in gray. **Figure 6.7d** is similar but for the inverted bi-objective relaxation in case LIE, and shows a reduced search space compared to **Figure 6.7c**.

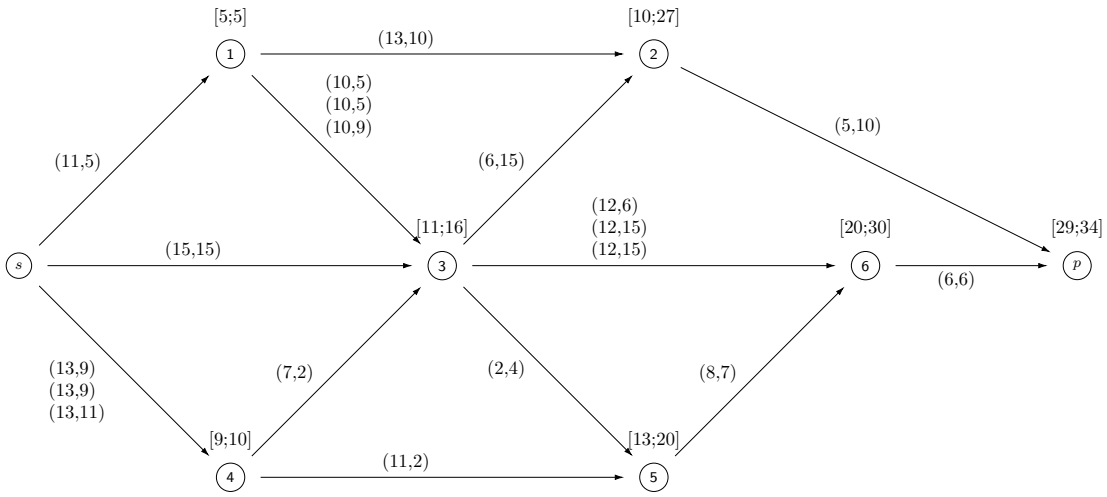


Figure 6.6: Three instances of the RWLPP

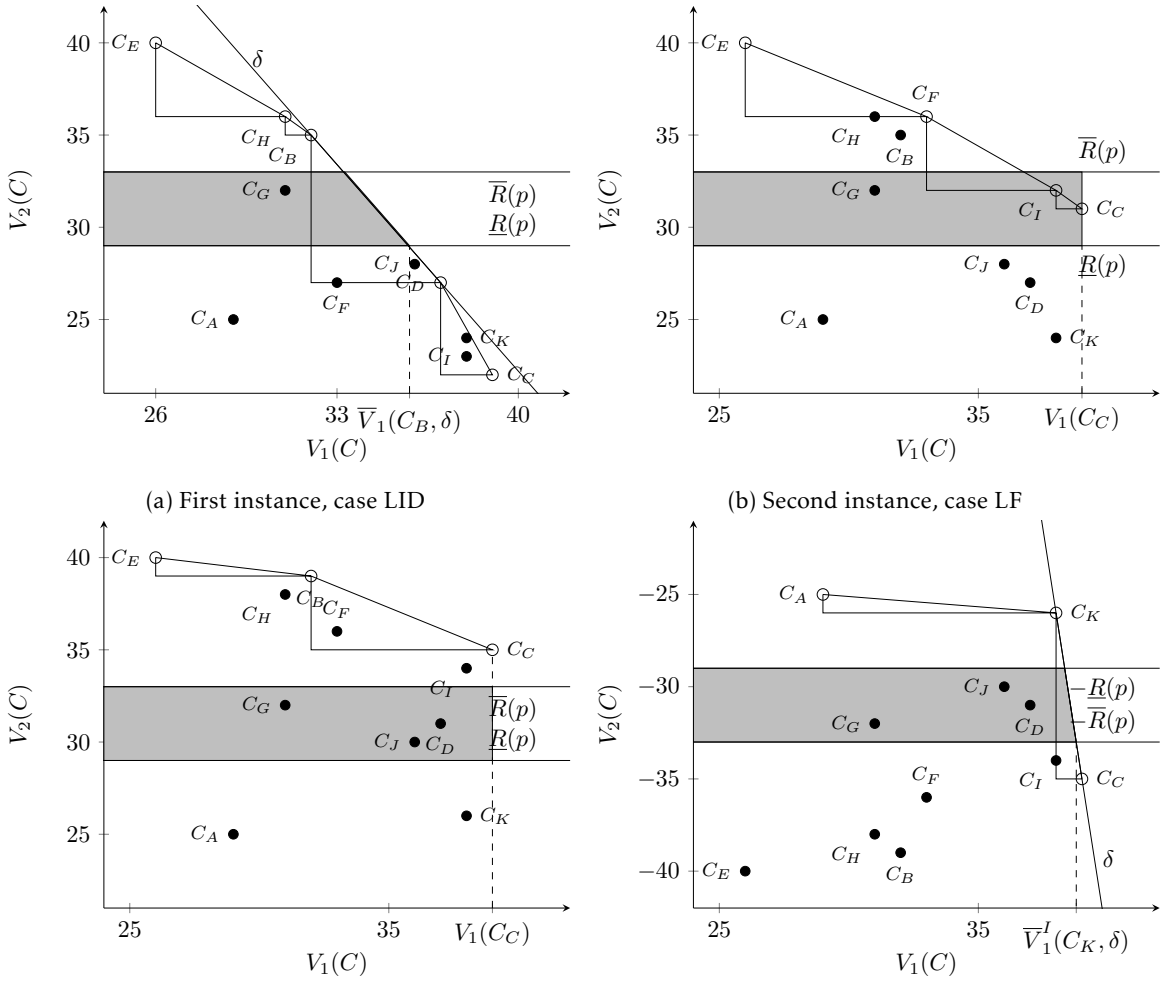
	First instance		second instance		third instance	
Path	V(.)	R(.)	V(.)	R(.)	V(.)	R(.)
$C_A = (1,2)$	29	25	29	25	29	25
$C_B = (1,3,2)$	32	35	32	35	32	39
$C_C = (1,3,6)$	39	22	39	31	39	35
$C_D = (1,3,5,6)$	37	27	37	27	37	31
$C_E = (3,2)$	26	40	26	40	26	40
$C_F = (3,6)$	33	27	33	36	33	36
$C_G = (3,5,6)$	31	32	31	32	31	32
$C_H = (4,3,2)$	31	36	31	36	31	38
$C_I = (4,3,6)$	38	23	38	32	38	34
$C_J = (4,3,5,6)$	36	28	36	28	36	30
$C_K = (4,5,6)$	38	24	38	24	38	26

Table 6.5: Value and resource of each path for each instance of the RWLPP

6.5.2 Bi-Objective relaxation of the Resource Windows algorithm

In this section, we present the Bi-Objective relaxation of Resource Windows (BORWin) algorithm, which aims to solve the RWLPP. To do so, a first phase consists in defining the region of the corresponding BOLPP, containing the optimal solution of the RWLPP. Then, a second phase aims to enumerate the solutions of this region, until the optimal solution of the RWLPP is found and proven optimal.

First phase. In the following, we explain **Algorithm 6** for the first phase. First we determine if the RWLPP instance is in case LF, LID or LIE, and the related search space. To do so, we compute a path C_1 maximizing $V_1(C_1)$ without taking into account the resource windows, which then means solving an LPP in polynomial time in graph G_C (line 1). In LF case, the first phase stops, path C_1 and vector $\delta = [1,0]$ are considered in the second phase. In LIE and LID cases, we consider the corresponding



(a) First instance, case LID

(b) Second instance, case LF

(c) Third instance, case LIE

(d) Third instance, case LIE, with negative resource

Figure 6.7: Representation of each path of the BOLPP in the objective space

BOLPP. As both cases are symmetric, in the following we only detail LID case. Now the aim is to find a pair of consecutive paths in P with their $V_2()$ values respectively above and below $\underline{R}(p)$. We know that C_1 maximizes $V_1(C_1)$, hence if we also consider C_2 maximizing $V_2(C_2)$ (line 14 and 15), then we have two Pareto-supported paths, with $\underline{R}(p) \in [V_2(C_1), V_2(C_2)]$. Consider vector $\delta = [1, \frac{V_1(C_1)-V_1(C_2)}{V_2(C_2)-V_2(C_1)}]$ (line 25), then $\mu_\delta(C_1) = \mu_\delta(C_2)$. Let C_3 be the path maximizing $\mu_\delta(C_3)$ (line 26). If $C_3 =_p C_1$ or $C_3 =_p C_2$, then path C_1 and vector $\delta = [1, \frac{V_1(C_1)-V_1(C_2)}{V_2(C_2)-V_2(C_1)}]$ are considered in the second phase. Otherwise, if $C_3 > \underline{R}(p)$, then one can consider pair (C_1, C_3), and if $C_3 \leq \underline{R}(p)$, one can consider pair (C_3, C_2) (lines 19 to 23). Vector δ is then recomputed, and a path maximizing the aggregated value until convergence is reached, i.e., no new path is obtained when maximizing the aggregated value. From **Corollary 2**, **Algorithm 6** ends. Also, it returns path C and vector δ minimizing $\overline{V}_1(C, \delta)$, as it relies on **Theorem 14** and **Property 43**.

Algorithm 6 BORWmin: first phase

Require: An RWLPP graph $G_C = (A_C, V_C)$
Compute C_1 maximizing $V(C_1)$
$caseLIE \leftarrow False$, $caseLID \leftarrow False$
if $V_2(C_1) \in [\underline{R}(p); \overline{R}(p)]$ **then**
    return C_1, $\delta = [1, 0]$
end if
if $V_2(C_1) < \underline{R}(p)$ **then**
    Consider the BOLPP with $V_1(C) = V(C)$ and $V_2(C) = R(C)$
    $caseLID \leftarrow True$
end if
if $V_2(C_1) > \overline{R}(p)$ **then**
    Consider the BOLPP with $V_1(C) = V(C)$ and $V_2(C) = -R(C)$
    $caseLIE \leftarrow True$
end if
$\delta \leftarrow [0, 1]$
Compute C_2 maximizing $\mu_\delta(C_2)$
$C_3 \leftarrow \emptyset$
while $C_3 \neq_p C_1$ and $C_3 \neq_p C_2$ **do**
    if $C_3 \neq \emptyset$ **then**
        if ($caseLIE$ and $V_2(C_3) > -\overline{R}(p)$) or ($caseLID$ and $V_2(C_3) > \underline{R}(p)$) **then**
            $C_2 \leftarrow C_3$
        else
            $C_1 \leftarrow C_3$
        end if
    end if
    $\delta \leftarrow [1, \frac{V_1(C_1) - V_1(C_2)}{V_2(C_2) - V_2(C_1)}]$
    Compute C_3 maximizing $\mu_\delta(C_3)$
end while
return C_1, δ

Second phase. In the following, let δ be the vector obtained from the first phase. The aim of the second phase is to enumerate the solutions of the search space defined from the first phase. To do so, we implicitly enumerate paths of the BOLPP with a variant of the K-best paths algorithm [62] for the LPP problem with objective $\mu_\delta(.)$. These paths are progressively modified in order to build feasible paths of the RWLPP that lie in the search space. In order to make such progressive modification, we introduce the concept of hybrid paths.

Definition 56 (*Hybrid path H*)

A hybrid path H is composed of two sub-paths: H_{RW} from s to a vertex v, and H_{BO} from v to p. Sub-path H_{RW} satisfies the resource windows of the RWLPP, while H_{BO} relaxes them.

From such hybrid paths, it is possible to define upper bounds of value $\mu_\delta(C)$ for any path C feasible for the RWLPP.

Theorem 15

Let $H = (H_{RW}, H_{BO})$ be a hybrid path from s to p, with H_{BO} maximizing $\mu_\delta(H_{BO})$. For any path C from s to p feasible for the RWLPP, starting with H_{RW}, then $\mu_\delta(H) \geq \mu_\delta(C)$.

Proof : By definition, H_{RW} is a path from s to a vertex v, and H_{BO} is from v to p. As C starts with H_{RW}, then it can be decomposed as follows: $C = (H_{RW}, C')$, with C' from v to p.

Suppose $\mu_\delta(H) < \mu_\delta(C)$, then as $H = (H_{RW}, H_{BO})$ and $C = (H_{RW}, C')$, we deduce $\mu_\delta(H_{BO}) < \mu_\delta(C')$. Recall that sub-path H_{BO} relaxes the resource windows, and C' takes them into account as C is feasible for the RWLPP. Hence, there is a contradiction, H_{BO} cannot maximize $\mu_\delta(H_{BO})$. ■

It is also possible to define a lower bound on the value $\mu_\delta(C^*)$, with C^* the optimal path of the RWLPP.

Theorem 16

Let C and C^* be respectively a feasible and the optimal path for the RWLPP. Let $\underline{\mu_\delta(C)} = \delta_1 \cdot V_1(C) + \delta_2 \cdot \underline{R}(p)$ in LID and LF cases and $\underline{\mu_\delta(C^*)} = \delta_1 \cdot V_1(C) + \delta_2 \cdot -\overline{R}(p)$ in LIE case. Then the optimal path C^* of the RWLPP is such that $\mu_\delta(C^*) \geq \underline{\mu_\delta(C)}$.

Proof : First consider cases LID and LF, i.e., $V_2(C^*) = R(C^*)$. Clearly, $V_1(C^*) \geq V_1(C)$, $V_2(C^*) \geq \underline{R}(p)$. Hence $\mu_\delta(C^*) \geq \underline{\mu_\delta(C)} = \delta_1 \cdot V_1(C) + \delta_2 \cdot \underline{R}(p)$.

Consider now case LIE, i.e., $V_2(C^*) = -R(C^*)$. Clearly, $V_1(C^*) \geq V_1(C)$ and $V_2(C^*) \geq -\overline{R}(p)$. Hence $\mu_\delta(C^*) \geq \underline{\mu_\delta(C)} = \delta_1 \cdot V_1(C) + \delta_2 \cdot -\overline{R}(p)$. ■

Algorithm 7 describes the enumeration of the second phase, which is as follows. We first compute a hybrid path $H = (H_{RW}, H_{BO})$ with $H_{RW} = \emptyset$ and H_{BO} the optimal path for the LPP from s to p with objective $\mu_\delta(.)$. Then, we consider a list of hybrid paths, initialized as $L = [H]$. List L will be kept sorted by decreasing value $\mu_\delta(.)$ during the whole algorithm. As long as L is not empty, we proceed with the following steps. We select H, the first hybrid path in L at line 5. If H is feasible, we obtain a lower bound $\underline{\mu_\delta(H)}$ on $\mu_\delta(C^*)$ as proven in **Theorem 16**. Hence, we remove from L all partial paths H' such that $\mu_\delta(H') < \underline{\mu_\delta(H)}$ at line 8. If H is not feasible, it is not feasible due to H_{BO} and we extend H_{RW}. Let $a = (u, v)$ be the first arc of H_{BO} from u to v. A classical labeling extension consists of building hybrid paths $H' = (H'_{RW}, H'_{BO})$ with $H'_{RW} = H_{RW} \cup \{(u, v')\}$, $v' \neq v$ and H'_{LW} the optimal path for the LPP from v' to p considering objective function $\mu_\delta(.)$. In this case, we generalize this extension in order to generate more new hybrid paths and obtain feasible paths quicker. This generalized extension described in the following corresponds to lines 10 to 20. Let $a = (u, v)$ be any arc of H_{BO}. Let SH be the subset of H_{BO}, containing all arcs before a. The extension consists of building $H' = (H'_{RW}, H'_{BO})$ with $H'_{RW} = H_{RW} \cup SH \cup \{u, v'\}$ and H'_{BO} the longest path for the LPP from v' to p with objective function $\mu_\delta(.)$. For each H' generated in this way, if H'_{RW} satisfies the resource windows, then H' is added to L. Otherwise, H' is discarded.

Algorithm 7 Second phase

Require: An RWLPP graph $G_C = (A_C, V_C)$ and a vector δ
Compute H_{BO} the shortest path from s to p maximizing $\mu_\delta(H_{BO})$
$H \leftarrow (\emptyset, H_{BO})$
$L \leftarrow [H]$
while $L \neq \emptyset$ **do**
    Select H the first element of L
    if H feasible for the RWLPP **then**
        compute the value $\underline{\mu_\delta(H)}$
        remove H' from L such that $\mu_\delta(H') < \underline{\mu_\delta(H)}$
    else
        for k in $[0; |H_{BO}| - 1]$ **do**
            get SH the k first arcs of H_{BO}
            get $a = (u, v)$ the $k+1^{\text{th}}$ arc of H_{BO}
            for arc $a' = (u, v') \in A_C$, $v' \neq v$ **do**
                $H'_{RW} = H_{RW} \cup SH \cup a'$
                compute H'_{BO} the path from v' to p maximizing $\mu_\delta(H'_{BO})$
                if $\mu_\delta(H) \geq \underline{\mu_\delta(C)}$ for any C feasible for the RWLPP **then**
                    add H to L while keeping L sorted by decreasing $\mu_\delta(.)$
                end if
            end for
        end for
    end if
end while

Further improvements. Note that this second phase can be further improved. In the following, we present three improvements that speed-up the algorithm for the 1-HUC^{DRM} problem.

A first improvement is in the case there exists an upper bound $h_1(C)$ for the value $V_1(C)$ of a feasible path. As soon as a feasible path C for the RWLPP is found, we can remove from L all hybrid path H for which $h_1(H) < V_1(C)$. In the case of the 1-HUC^{DRM} problem, such upper bound can be obtained with **Algorithm 4**, defined in **Section 6.4.1**, featured in the HA* algorithm.

The second improvement is to use a classical dominance rule for the RCSPP at each vertex. Let H_{RW} be a partial path from s to v. Any partial path H'_{RW} from s to v such that $V_2(H'_{RW}) = V_2(H_{RW})$ and $V_1(H'_{RW}) \geq V_1(H_{RW})$ cannot yield the optimal solution of the RWLPP. Indeed, both partial path use the exact same resource, while H_{RW} has a higher value. Hence, H_{RW} dominates H'_{RW}. The proposed implementation is to store for each node v a list containing the pairs of values $(V_1(C), V_2(C))$ of all partial path enumerated from s to v. This implementation is particularly effective if there are many paths using the exact same amount of resource between node s and a node v, as the lists at each vertex will remain small. This is the case for the 1-HUC^{DRM} problem, as the operating points are the same, the exact same set of water flows is used at each time period. Note however that this implementation can produce very large lists at each node, which can slow down BORWin algorithm for RWLPP instances for which there is no such structure for the use of the resource.

The third improvement consists in storing the optimal solution of the LPP from any vertex v to the target vertex p. Indeed, for two partial paths H_{RW} and H'_{RW} from s to v, the corresponding H_{BO} is the same for both. As such, it is possible to only compute H_{BO} between v to p once.

6.5.3 Experimental results

All presented results are computed on a single thread of an Intel Core i7-9850H CPU @ 2.60GHz processor of 12 cores, with Linux as operating system. All algorithms are developed with C++ and version 12.8 of CPLEX is used.

For this study, we compare four approaches: model $M_{op\text{-}DRM}$ solved by CPLEX, RCSPP algorithm as described in [2], HA* (see **Section 6.4**) and BORWin with the improvements dedicated to the 1-HUC^{DRM} problem. For this comparison, we consider a first set of 83 EDF instances. These instances have been slightly modified, in order for at least either the upper bound or the lower bound on the resource to be active for at least one time period. To do so, we applied a few modifications if necessary on the parameters of the upstream reservoir: the initial volume V_0^1, bounds at the last time period $\underline{V}_T^1$, $\overline{V}_T^1$ or the additional intake of water in the reservoirs A_t^1. Note that these instances remain realistic as no modification have been made on the data describing actual plant or the reservoirs. In particular, the modified parameters are part of the parameters that change from one instance to another for a given plant.

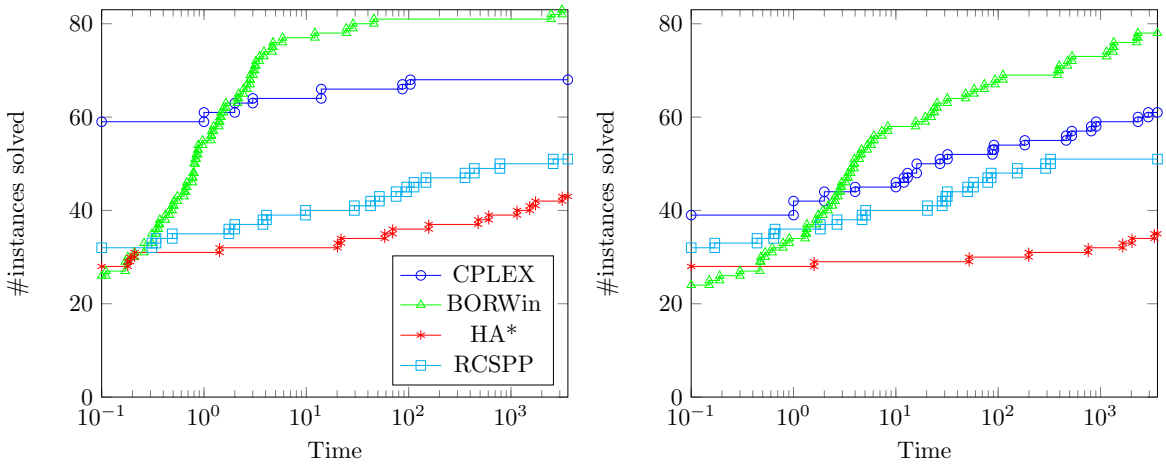
Preliminary results have shown that when the price of the energy Λ_t is very similar from one time period to another, instances of the 1-HUC^{DRM} are harder to solve. As mentioned in **Section 3.2**, the resource windows can be interpreted as nested knaspack and covering inequalities. These preliminary results are consistent with the observations in **Section 4.7**, where instances of the Knapsack Problem are harder when their values are highly correlated to their weight [83]. As such, we also consider a second set of 83 instances, built as follows. For each instance of the first set, an instance of the second set is created with Λ_t being a random value in $[0.95 \cdot \Lambda_1; 1.05 \cdot \Lambda_1]$. For the following, the first set is denoted as the set C, and the second set as the set D.

Figure 6.8a and **6.8b** show, for each approach, the number of instances solved with respect to the time for instance set C. **Figure 6.8b** is similar to **Figure 6.8a** but for instance set D. **Table 6.6** shows the value of the solutions and the time required for each approach for a short list of instances, where each selected instance is labeled from its number and set. For CPLEX, we also provide #node the number of nodes developed as well as the optimality gap, being opt (resp inf, sub) if the solution returned is optimal (resp. infeasible, suboptimal). We also added #iter the number of iterations of the second phase for BORWin. The complete tables are provided in **Appendix C.1**.

Clearly, these figures show that HA* and the RCSPP algorithms are not suitable to solve the 1-HUC^{DRM} problem. On the contrary, BORWin appears to be very well suited as it is the most efficient approach as soon as the computational times exceed 3 seconds. Moreover, BORWin solved 15 more instances in

each set than the MILP which is the current approach at EDF. These results also confirm the preliminary results, as it appears that instances of set D are harder to solve than the ones of set C.

As stated in **Section 6.4.3**, HA* algorithm is not suitable due to its bound, which does not take into account ramping or min-up/down constraints. The RCSPP algorithm is inefficient as the dominance rule is seriously weakened due to the lower bound, as stated in **Section 6.2.2**. When it comes to solving the MILP, there are 12 instances of the first set, and 5 of the second set, where the solution returned is infeasible due to numerical errors, as it violates one of the resource window. This violation is in most cases below 0.1%, but can reach up to 10%. For instance 64-C in **Table 6.6**, the volume of the upstream reservoir violates all upper bound from time period 29 to 96. At time period T, the volume reaches 29175.04 whereas the upper bound $\overline{V}^1_T$ is 26380. Such numerical errors have been observed, and a study has shown that these errors are due to floatting-point precision of the solver [14,91]. In addition to the infeasible solutions, there are 2 instances of the first set, and 6 of the second set where the instances are suboptimal. Instance 48-D shows a case where the value of solution obtained when solving the MILP model is 0.009% below the value of the optimal solution obtained with BORWin. The gap between the value of these solutions and the optimal value of the optimal solution obtained with BORWin is at most 0.01%. This gap is consistent with the optimality gap of CPLEX, which is 10^{-4} by default. Note that it is also possible that solving the MILP model yields a solution that is suboptimal and also infeasible. This is the case for instance 57-C.



(a) Results for set C

(b) Results for set D

Figure 6.8: Number of instances solved by each approach with respect to the time for each instance set

6.6 Conclusion

In this section, two graph representations are presented for the discretized 1-HUC problem as a special case of the RWSPP. We distinguish the benefits and the drawback of each of these graphs: one

	CPLEX				RCSPP		HA*		BORWin		
instance	value	gap	time	#nodes	value	time	value	time	value	#iter	time
57-C	3 931 900.0	inf	0.0	1177	-	3716.47	-	3600.29	3 931 920.0	287191	2468.04
64-C	95 093.1	inf	0.0	0	-	3632.43	-	3600.61	95 093.1	286	0.53
48-D	452 277.0	0.04	3599.0	5319419	-	3625.98	-	3600.47	452 318.0	123282	2254.47

Table 6.6: Performance of model $M_{op\text{-}DRM}$ solved with CPLEX, the RCSPP algorithm, HA* and BORWin on EDF inspired instances 1 to 42

can easily take into account resource windows, but requires an exponential number of vertices; the other does not take into account resource windows but features a polynomial number of vertices. For each of these graph representations, an algorithm is proposed.

First, the HA* algorithm has been proposed. This algorithm is an exact variant of the A* algorithm, with a dual bound specific to the discretized 1-HUC problem without ramping nor min-up/down constraints. Experimental results showed that HA* is more stable with respect to the computational time against the current approach at EDF and an RCSPP algorithm for instances without ramping nor min-up/down constraints. However, the current state of HA* does not handle efficiently additional constraints of the discretized 1-HUC problem.

Then, we introduced algorithm BORWin, which computes the optimal solution of an RWSPP through a bi-objective relaxation of the resource windows. Inspired by a standard two-phase algorithm for bi-objective problems, combined with resource windows, a reduced search space has been exhibited in which BORWin implicitly enumerates solutions. As BORWin can be used to solve any RWSPP, it can take into account the additional constraints of the discretized 1-HUC problem. The ramping, min-up and min-down constraints are directly tackled by construction of the compact graph. On a large set of EDF instances, BORWin outperforms HA*, the RCSPP algorithm as well as the MILP approach currently exploited at EDF. Besides, these experimental results also confirm that solving a MILP model, in particular for the HUC problem, can produce significant numerical errors, yielding infeasible or suboptimal solutions.

These good results allow to consider the use of BORWin at EDF. First it could be used to solve the 1-HUC^{DRM} problem, but could also be integrated in a decomposition method, such as the one defined in [3] to solve the HUC problem. This could yield an appealing alternative to the MILP approach. Interestingly, the upper bound obtained through the first phase of BORWin could be equivalent to the optimal value of the Dantzig Wolfe relaxation. One interesting perspective would be to study the link between these two bounds.

In the following chapter, we draw concluding remarks on the work detailed in this thesis. Moreover, we describe various interesting perspectives to extend our studies.

7 Conclusion

In this thesis, we studied modeling alternatives and algorithms to solve the single plant Hydro Unit Commitment (1-HUC) problem. The aim was to propose a performant solution approach for this problem.

7.1 Summary of contributions

In the general introduction (**Chapter 1**), we presented the context revolving around the 1-HUC problem and the importance of solving it efficiently when scheduling the short-term production at EDF. Then, we defined a generic 1-HUC problem, with its hydraulic constraints and its main non-linearities. The hydraulic constraints include the bound on the reservoirs volume forming resource windows, the ramping constraints and the min-up/down constraints. The considered non-linearities are the power function, as a non-convex non-concave function of the water flow and the water head, the latter being a non-linear function of the reservoir volumes.

In **Chapter 2**, we compared modeling alternatives for the 1-HUC problem, focusing on the power function. For this purpose, we considered a simplified non-linear model, leaving aside ramping and min-up/down constraints. This non-linear model is defined for the general case, but also for the fixed-head case, which makes sense is some situations encountered in practice. We provided seven different modeling alternatives, covering a wide set of non-linear model families. In addition, we considered five different non-linear off-the-shelf solvers, as each solver implements its own non-linear tools. From this comparison of modeling alternatives, we highlighted the most efficient models, in terms of computational time, precision and feasibility, for both the general case and the fixed-head case. For each of them, we also indicated which solver is the most suitable, leading to overall best performances.

In **Chapter 3**, we selected one of the highlighted model from the previous comparison, to be the one studied in the remainder of the thesis in order to study the hydraulic constraints. More precisely, the model selected is based on a discrete set of flows for the fixed-head 1-HUC. In such a case, the model features operating points, an operating point being a pair of a flow and the associated produced power. This model has been rewritten so that the resource windows on the volume become resource windows on the cumulated flow, i.e., the sum of the flow since the first time period. For this model, we defined a polynomial time bound-tightening technique. Finally, we specified the resulting discretized 1-HUC problems and their formulations studied in the remainder of the thesis.

In **Chapter 4**, we conducted a polyhedral study of the Symmetric-weight Chain Precedence Knap-

sack Problem (SCPKP) and proposed a two-phase Branch & Cut algorithm to solve it. The rationale behind studying the SCPKP is that this problem can be specifically defined as the combinatorial core of the discretized 1-HUC problem. For this problem, we defined a pattern structure able to handle polyhedral symmetries, i.e., when the symmetries of a solution yield feasible solutions, but with different values. Moreover, we exhibited the pattern inequalities associated to the patterns. Necessary facet-defining conditions have been introduced and were proven to be sufficient for a family of patterns. We then presented a two-phase Branch & Cut algorithm, relying on these patterns. The first phase aims at generating patterns verifying necessary facet-defining conditions, the second phase consists in using these patterns in a Branch & Cut scheme. This algorithm has been compared to state-of-the-art Branch & Cut algorithms, to show its better performances.

In **Chapter 5**, we extended the polyhedral results of the SCPKP to the discretized 1-HUC problem without ramping nor min-up/down constraints. For this purpose, we introduced the Inverted SCPKP (ISCPKP), so that each window resource of the discretized 1-HUC problem is captured by a pair (SCPKP, ISCPKP). We first show that the ISCPKP can be rewritten as an SCPKP. Consequently, all results for the SCPKP translate to the ISCPKP. Moreover, we demonstrated that only a polynomial number of (I)SCPKP are necessary to study the polyhedron of the discretized 1-HUC problem. As the polyhedral symmetries of the SCPKP do not hold completely for the discretized 1-HUC problem, we extended the definition of the patterns and the necessary facet-defining conditions. Finally, we sketched the extension of the two-phase Branch & Cut algorithm to this discretized 1-HUC problem, by introducing few pre-processing steps.

In **Chapter 6**, we revisited the discretized 1-HUC problem as a Longest Path Problem with Resource Windows (RWLPP) in a graph. More precisely, two different graphs are presented, a cumulated-flow-expanded one and a compact one. For the first one, we presented HA*, an exact variant of the A* algorithm, featuring a dedicated dual bound. For the second one, we presented BORWin, a two-phase algorithm based on the bi-objective relaxation of the window constraints. The algorithm is inspired by the standard two-phase algorithm for bi-objective optimization, but it takes advantage of the resource windows to further restrict the search space. We compare the performances of our algorithms against two state-of-the-art approaches. For instances without ramping nor min-up/down constraints, these results show that HA* is particularly efficient against two state-of-the-art approaches. The BORWin algorithm outperforms all other approaches for the general case. These good results allow to consider a future deployment of BORWin in production at EDF to replace the MILP-based approaches used so far.

7.2 Software and publications

The successful completion of this thesis was dependent on a substantial amount of computer developments. For each chapter, a functional corresponding code has been developed. Besides, we also redeveloped all state-of-the-art algorithms used in our experimental results except, of course, the tools

already present in the solvers, such as the Branch & Cut framework of CPLEX. Most of the codes have been written in C++. The exception being the instance generators developed in python, which write instances files either in text or GAMS format, as well as calls to non-linear solvers available on NEOS Server, which were also written in python. All codes developed during this thesis will be delivered to EDF for future research, or in the case of the BORWin algorithm to be tested in their industrial environment. The instances of the 1-Hydro Unit Commitment problems have not been made publicly available due to their strong resemblance to real EDF instances. In contrast, the Symmetric-weight Chain Precedence Knapsack Problem instances have been made publicly available online.

The research work developed in this thesis has already led to a paper currently under review (third round) for the international journal Computers & Operations Research and a paper published in the proceedings of an international conference (FedCIS 2023). One submission is currently under review for a conference with published proceedings (IOS 2024). Another paper is currently at its final writing state, and will be submitted to an international journal. The work has also been presented during an international conference without proceedings (ISCO 2022) and four national conferences (JPOC 2023 and ROADEF 2021-2022-2023). Finally, the algorithm based on a bi-objective relaxation of the window constraints from **Chapter 6** has been selected by the jury of the best student paper award for ROADEF 2024, as one of the finalists. All finalists will present their work in a dedicated session of the conference, after which the jury will announce the final results of the competition.

7.3 Perspectives

In the following, we present the perspectives corresponding to individual contributions presented in each chapter of this thesis. Then, we present two more general perspectives.

One could extend the model comparison from **Chapter 2**, by comparing refined versions of the highlighted models. As such, one could reveal the most relevant one for solving larger instances or instances taking into account a larger set of constraints. For instance, models M_{PWL}, $M_{2D\text{-}poly}$ and M_{op} can be enhanced with logarithmic disjunctive constraints [104]. The solvers used in our study feature bound-tightening techniques, but one could also consider a dedicated bound-tightening technique for each model. Furthermore, model M_{PWL} could be improved by optimizing the number of breakpoints with a guaranteed precision [76] and using PWL models from the literature. In particular, the formulation used is the so-called non-ideal formulation defined in [54]. A logarithmic formulation, proven to be ideal, could be used instead. Finally, models M_{bilin} and $M_{2D\text{-}poly}$ feature respectively bilinear and quadratic constraints. These are common non-linear functions for which there are dedicated programming techniques that could be applied.

In **Chapter 3** we defined a polynomial-time bound-tightening procedure for the 1-HUC^{DRM} problem, whereas in **Chapter 5** the bound-tightening technique for the 1-HUC^{D} problem requires to solve a MILP which does not take into account ramping nor min-up/down constraints. Both procedures do not yield optimal bounds for the 1-HUC^{DRM} problem, and the complexity of the MILP-based approach is

unknown. Therefore, one perspective would be to study the complexity of the MILP-based approach, or to provide new procedures able to yield tighter bounds than the ones proposed.

In **Chapter 4**, we made a conjecture concerning the NP-hardness of the separation of pattern inequalities. A perspective would be to verify if this conjecture is valid. One could also study facet-defining inequalities of the Symmetric-weight Chain Precedence Knapsack Problem (SCPKP) with integer coefficients. This would complement the binary facet-defining inequalities identified in this thesis.

The results of the study of the SCPKP in **Chapter 4** have been extended to the 1-HUC^D problem in **Chapter 5**. However, the polyhedral symmetry aspect of the SCPKP partially extends to the 1-HUC^D problem. As such, future work should examine asymmetric aspects of the 1-HUC^D problem in order to enhance the pattern inequalities. Even if we did extend the two-phase Branch & Cut algorithm to the 1-HUC^D problem, the time constraints of the thesis did not allow for computational experiments. One could compare this algorithm numerically to state-of-the-art algorithms to show its performances. More generally, one could extend the two-phase Branch & Cut algorithm to other problems featuring polyhedral symmetries.

In **Chapter 6**, we detailed two graph-based algorithms, the HA* algorithm and the BORWin algorithm, respectively. One perspective would be to extend the HA* algorithm in order to take into account the additional constraints when computing the dual bound. As such, both HA* and BORWin would be efficient to solve the 1-HUC^{DRM} problem. This perspective is particularly relevant if one identifies cases where one of these algorithm outperforms the other. The good results for BORWin algorithm bode well for a broader use of BORWin at EDF in the very near future. First, it could readily be used in place of the MILP solving for valleys with a single plant. Second, beyond the 1-Hydro Unit Commitment use case, BORWin appears to be well suited for other use cases where window constraints on one resource arise. For instance, with the emergence of fluctuating renewable electricity sources such as wind and solar, the need for flexibilities both on the generation side and on the demand side opens up new perspectives for BORWin, in particular for controlling demand response of thermostatic loads [99], e.g., water-heaters, fridges, heating, cooling, or even electric vehicle charging.

Regarding more general perspectives, one of them would be to combine the work presented in different chapters. More precisely, one could create a hybrid method, featuring the results of the polyhedral study and the graph algorithms. The idea would be either to use graph algorithms to provide bounds for the two-phase Branch & Cut, or to use patterns to enhance the graph algorithms.

As results showed that BORWin algorithm is effective to solve the 1-Hydro Unit Commitment problem, one could use it in a decomposition framework, such as the one described in [3]. Doing so could lead to an approach able to solve the Hydro Unit Commitment problem defined on valleys with multiple plants more efficiently than current approaches.

Bibliography

1. Abu Al-Haija Q. A stochastic estimation framework for yearly evolution of worldwide electricity consumption. Forecasting 2021;3:256–66.
2. Ackooij W van, d'Ambrosio C, Liberti L, Taktak R, Thomopulos D, and Toubaline S. Shortest path problem variants for the hydro unit commitment problem. Electronic Notes in Discrete Mathematics 2018;69:309–16.
3. Ackooij W van, d'Ambrosio C, Thomopulos D, and Trindade RS. Decomposition and shortest path problem formulation for solving the hydro unit commitment and scheduling in a hydro valley. European Journal of Operational Research 2021;291:935–43.
4. Amani A and Alizadeh H. Solving hydropower unit commitment problem using a novel sequential mixed integer linear programming approach. Water Resources Management 2021;35:1711–29.
5. Arce A, Ohishi T, and Soares S. Optimal dispatch of generating units of the Itaipu hydroelectric plant. IEEE Transactions on power systems 2002;17:154–8.
6. Ascheuer N, Fischetti M, and Grötschel M. A polyhedral study of the asymmetric traveling salesman problem with time windows. Networks: An International Journal 2000;36:69–79.
7. Balas E. Facets of the knapsack polytope. Mathematical programming 1975;8:146–64.
8. Balas E, Fischetti M, and Pulleyblank WR. The precedence-constrained asymmetric traveling salesman polytope. Mathematical programming 1995;68:241–65.
9. Barrett C, Bisset K, Holzer M, Konjevod G, Marathe M, and Wagner D. Engineering label-constrained shortest-path algorithms. In: *Algorithmic Aspects in Information and Management: 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008. Proceedings 4*. Springer. 2008:27–37.
10. Beasley JE and Christofides N. An algorithm for the resource constrained shortest path problem. Networks 1989;19:379–94.
11. Bellman R. On a routing problem. Quarterly of applied mathematics 1958;16:87–90.
12. Belotti P. Couenne: a user's manual. Tech. rep. Lehigh University, 2009.
13. Bendotti P, Fouilhoux P, and Rottner C. The min-up/min-down unit commitment polytope. Journal of Combinatorial Optimization 2018;36:1024–58.

14. Bendotti P and Févotte F. Impact de l'arithmétique flottante sur la résolution de programmes linéaires. In: *10èmes Journées Polyèdres et Optimisation Combinatoire*. 2016.
15. Boland N, Bley A, Fricke C, Froyland G, and Sotirov R. Clique-based facets for the precedence constrained knapsack problem. Mathematical programming 2012;133:481–511.
16. Borghetti A, D'Ambrosio C, Lodi A, and Martello S. An MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. IEEE Transactions on power systems 2008;23:1115–24.
17. Boyd E. Polyhedral results for the precedence-constrained knapsack problem. Discrete Applied Mathematics 1993;41:185–201.
18. Brandao LC, Pessanha JF, Khenayfis LS, Diniz AL, Pereira RJC, and Junior CAA. A Data-Driven Representation of Aggregate Efficiency Curves of Hydro Units for the Mid-Term Hydrothermal Coordination Problem. Electric Power Systems Research 2022;212:108511.
19. Catalão JPdS, Pousinho HMI, and Mendes VMF. Mixed-integer nonlinear approach for the optimal scheduling of a head-dependent hydro chain. Electric Power Systems Research 2010;80:935–42.
20. Ceria S, Cordier C, Marchand H, and Wolsey LA. Cutting planes for integer programs with general integer variables. Mathematical programming 1998;81:201–14.
21. Chabane B, Basseur M, and Hao JK. Lorenz dominance based algorithms to solve a practical multiobjective problem. Computers & Operations Research 2019;104:1–14.
22. Chakhchoukh Y, Panciatici P, and Mili L. Electric Load Forecasting Based on Statistical Robust Methods. Power Systems, IEEE Transactions on 2011;26:982 –991.
23. Cheng CT, Liao SL, Tang ZT, and Zhao MY. Comparison of particle swarm optimization and dynamic programming for large scale hydro unit load dispatch. Energy Conversion and Management 2009;50:3007–14.
24. Christof T and Löbel A. Polyhedron representation transformation algorithm. URL: https://porta.zib.de 2009.
25. Conejo AJ, Arroyo JM, Contreras J, and Villamor FA. Self-scheduling of a hydro producer in a pool-based electricity market. IEEE Transactions on power systems 2002;17:1265–72.
26. Cplex II. V12. 1: User's Manual for CPLEX. International Business Machines Corporation 2009;46:157.
27. Croxton KL, Gendron B, and Magnanti TL. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. Management Science 2003;49:1268–73.
28. Czyzyk J, Mesnier MP, and Moré JJ. The NEOS server. IEEE Computational Science and Engineering 1998;5:68–75.
29. Damcı-Kurt P, Küçükyavuz S, Rajan D, and Atamtürk A. A polyhedral study of production ramping. Mathematical Programming 2016;158:175–205.

30. Del Pia A, Linderoth J, and Zhu H. On the complexity of separating cutting planes for the knapsack polytope. Mathematical Programming 2023:1–27.

31. Dijkstra EW. A note on two problems in connexion with graphs. Numerische mathematik 1959;1:269–71.

32. D'Ambrosio C, Lodi A, and Martello S. Piecewise linear approximation of functions of two variables in MILP models. Operations Research Letters 2010;38:39–46.

33. Edmonds J and Karp RM. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM) 1972;19:248–64.

34. Edom E, Anjos M, D'Ambrosio C, Van Ackooij W, Côté P, and Séguin S. On the impact of the power production function approximation on hydropower maintenance scheduling. Les Cahiers du GERAD G–2020–22, GERAD 2020.

35. Ehlers R. Computing the complete pareto front. arXiv preprint arXiv:1512.05207 2015.

36. Eppstein D. Finding the k shortest paths. SIAM Journal on computing 1998;28:652–73.

37. Espinoza D, Goycoolea M, and Moreno E. The precedence constrained knapsack problem: Separating maximally violated inequalities. Discrete Applied Mathematics 2015;194:65–80.

38. Fan W, Guan X, and Zhai Q. A new method for unit commitment with ramping constraints. Electric Power Systems Research 2002;62:215–24.

39. Ferreira CE, Martin A, and Weismantel R. Solving multiple knapsack problems by cutting planes. SIAM Journal on Optimization 1996;6:858–77.

40. Finardi E, Takigawa F, and Brito B. Assessing solution quality and computational performance in the hydro unit commitment problem considering different mathematical programming approaches. Electric Power Systems Research 2016;136:212–22.

41. Finardi EC and Scuzziato MR. Hydro unit commitment and loading problem for day-ahead operation planning problem. International Journal of Electrical Power & Energy Systems 2013;44:7–16.

42. Finardi EC and Silva EL da. Solving the hydro unit commitment problem via dual decomposition and sequential quadratic programming. IEEE transactions on Power Systems 2006;21:835–44.

43. García-González J, Parrilla E, Barquín J, Alonso J, Sáiz-Chicharro A, and González A. Underrelaxed iterative procedure for feasible short-term scheduling of a hydro chain. In: *2003 IEEE Bologna Power Tech Conference Proceedings*, vol. 2. 2003:6 pp.

44. Garey MR and Johnson DS. Computers and intractability. A guide to the theory of NP-completeness 1979.

45. Geißler B, Martin A, Morsi A, and Schewe L. Using piecewise linear functions for solving MINLPs. In: *Mixed integer nonlinear programming*. Vol. 154. The IMA Volumes in Mathematics and its Applications. Springer, 2012:287–314.

46. Glasnovic Z and Margeta J. The features of sustainable solar hydroelectric power plant. Renewable energy 2009;34:1742–51.

47. Gomory RE. Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem. 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art 2010:77–103.

48. Gottschalk PG and Dunn JR. The five-parameter logistic: a characterization and comparison with the four-parameter logistic. Analytical biochemistry 2005;343:54–65.

49. Grabisch M and Labreuche C. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. Annals of Operations Research 2010;175:247–86.

50. Hart PE, Nilsson NJ, and Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 1968;4:100–7.

51. Hechme-Doukopoulos G, Brignol-Charousset S, Malick J, and Lemaréchal C. The short-term electricity production management problem at EDF. Optima Newsletter 2010;84:2–6.

52. Heintzmann A, Artigues C, Bendotti P, Ngueveu SU, and Rottner C. Efficient exact A* algorithm for the single plant Hydro Unit Commitment problem. In: *Proceedings of the 18th Conference on Computer Science and Intelligence Systems*. Vol. 35. Annals of Computer Science and Information Systems. IEEE, 2023:533–543. DOI: 10.15439/2023F5158.

53. Hojny C, Gally T, Habeck O, Lüthen H, Matter F, Pfetsch ME, and Schmitt A. Knapsack polytopes: a survey. Annals of Operations Research 2020;292:469–517.

54. Huchette J and Vielma JP. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. Operations Research 2023;71:1835–56.

55. IEA. Electricity Information: Overview. https://www.iea.org/reports/electricity-information-overview. Accessed: 2024-01-08. 2021.

56. Irnich S and Desaulniers G. Shortest path problems with resource constraints. Springer, 2005.

57. Johnston RE and Khan LR. Bounds for nested knapsack problems. European Journal of Operational Research 1995;81:154–65.

58. Kellerer H, Pferschy U, Pisinger D, Kellerer H, Pferschy U, and Pisinger D. The multiple-choice knapsack problem. Knapsack Problems 2004:317–47.

59. Kerschke P, Kotthoff L, Bossek J, Hoos HH, and Trautmann H. Leveraging TSP solver complementarity through machine learning. Evolutionary computation 2018;26:597–620.

60. Kruber M, Parmentier A, and Benchimol P. Resource constrained shortest path algorithm for EDF short-term thermal production planning problem. 2018. DOI: 10.48550/ARXIV.1809.00548. URL: https://arxiv.org/abs/1809.00548.

61. Kuhn HW. The Hungarian method for the assignment problem. Naval research logistics quarterly 1955;2:83–97.

62. Lawler EL. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. Management science 1972;18:401–5.

63. Lee J, Leung J, and Margot F. Min-up/min-down polytopes. Discrete Optimization 2004;1:77–85.

64. Leensel RL van de, Van Hoesel C, and Klundert J Van de. Lifting valid inequalities for the precedence constrained knapsack problem. Mathematical programming 1999;86:161–85.

65. Letchford AN and Souli G. Lifting the knapsack cover inequalities for the knapsack polytope. Operations Research Letters 2020;48:607–11.

66. Li X, Li T, Wei J, Wang G, and Yeh WWG. Hydro unit commitment via mixed integer linear programming: A case study of the Three Gorges project, China. IEEE Transactions on Power Systems 2013;29:1232–41.

67. Lima RM, Marcovecchio MG, Novais AQ, and Grossmann IE. On the computational studies of deterministic global optimization of head dependent short-term hydro scheduling. IEEE Transactions on Power Systems 2013;28:4336–47.

68. Lin Y and Schrage L. The global solver in the LINDO API. Optimization Methods & Software 2009;24:657–68.

69. Margot F. Symmetry in integer linear programming. 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art 2009:647–86.

70. Mariano S, Catalão J, Mendes V, and Ferreira L. Optimising power generation efficiency for head-sensitive cascaded reservoirs in a competitive electricity market. International Journal of Electrical Power & Energy Systems 2008;30:125–33.

71. Markowitz HM and Manne AS. On the solution of discrete programming problems. Econometrica: journal of the Econometric Society 1957:84–110.

72. Miettinen K, Ruiz F, and Wierzbicki AP. Introduction to multiobjective optimization: interactive approaches. Multiobjective optimization: interactive and evolutionary approaches 2008:27–57.

73. Misener R and Floudas CA. ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. Journal of Global Optimization 2014;59:503–26.

74. Morales-España G, Gentile C, and Ramos A. Tight MIP formulations of the power-based unit commitment problem. OR spectrum 2015;37:929–50.

75. Moreno E, Espinoza D, and Goycoolea M. Large-scale multi-period precedence constrained knapsack problem: a mining application. Electronic notes in discrete mathematics 2010;36:407–14.

76. Ngueveu SU. Piecewise linear bounding of univariate nonlinear functions and resulting mixed integer linear programming-based solution methods. European Journal of Operational Research 2019;275:1058–71.

77. Orero S and Irving M. A genetic algorithm modelling framework and solution technique for short term optimal hydrothermal scheduling. IEEE Transactions on Power Systems 1998;13:501–18.

78. Pan K and Guan Y. A polyhedral study of the integrated minimum-up/-down time and ramping polytope. arXiv preprint arXiv:1604.02184 2016.

79. Paredes M, Martins L, and Soares S. Using semidefinite relaxation to solve the day-ahead hydro unit commitment problem. IEEE Transactions on Power Systems 2014;30:2695–705.

80. Park K and Park S. Lifting cover inequalities for the precedence-constrained knapsack problem. Discrete Applied Mathematics 1997;72:219–41.

81. Pérez JI and Wilhelmi JR. Nonlinear self-scheduling of a single unit small hydro plant in the day-ahead electricity market. Proc. of ICREPQ'07 2007.

82. Pérez-Díaz JI, Wilhelmi JR, and Arévalo LA. Optimal short-term operation schedule of a hydropower plant in a competitive electricity market. Energy Conversion and Management 2010;51:2955–66.

83. Pisinger D. Where are the hard knapsack problems? Computers & Operations Research 2005;32:2271–84.

84. Quesada I and Grossmann IE. Global optimization of bilinear process networks with multicomponent flows. Computers & Chemical Engineering 1995;19:1219–42.

85. Rajan D, Takriti S, et al. Minimum up/down polytopes of the unit commitment problem with start-up costs. IBM Res. Rep 2005;23628:1–14.

86. Ralphs TK, Saltzman MJ, and Wiecek MM. An improved algorithm for solving biobjective integer programs. Annals of Operations Research 2006;147:43–70.

87. Renaud A. Daily generation management at Electricité de France: from planning towards real time. IEEE Transactions on Automatic Control 1993;38:1080–93.

88. Ribeiro CC and Minoux M. A heuristic approach to hard constrained shortest path problems. Discrete Applied Mathematics 1985;10:125–37.

89. Rodriguez JA, Anjos MF, Côté P, and Desaulniers G. MILP formulations for generator maintenance scheduling in hydropower systems. IEEE Transactions on Power Systems 2018;33:6171–80.

90. Réseau-Energie O. Equilibre national mensuel production = consommation brute. https://odre.opendatasoft.com/explore/dataset/equilibre-national-mensuel-prod-conso-brute/information/. Accessed: 2024-01-08. 2023.

91. Sahraoui Y, Bendotti P, and d'Ambrosio C. Real-world hydro-power unit-commitment: Dealing with numerical errors and feasibility issues. Energy 2019;184:91–104.

92. Salem MB, Taktak R, Mahjoub AR, and Ben-Abdallah H. Optimization algorithms for the disjunctively constrained knapsack problem. Soft Computing 2018;22:2025–43.

93. Séguin S, Côté P, and Audet C. Self-scheduling short-term unit commitment and loading problem. IEEE Transactions on Power Systems 2015;31:133–42.

94. Skjelbred HI, Kong J, and Fosso OB. Dynamic incorporation of nonlinearity into MILP formulation for short-term hydro scheduling. International journal of electrical power & energy systems 2020;116:105530.

95. Smith EM and Pantelides CC. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. Computers & Chemical Engineering 1999;23:457–78.

96. Souza HG e, Finardi EC, Brito BH, and Takigawa FYK. Partitioning approach based on convex hull and multiple choice for solving hydro unit-commitment problems. Electric Power Systems Research 2022;211:108285.

97. Taktak R and D'Ambrosio C. An overview on mathematical programming approaches for the deterministic unit commitment problem in hydro valleys. Energy Systems 2017;8:57–79.

98. Tawarmalani M and Sahinidis NV. A polyhedral branch-and-cut approach to global optimization. Mathematical Programming 2 2005;103:225–49.

99. Tindemans SH, Trovato V, and Strbac G. Decentralized control of thermostatic loads for flexible demand response. IEEE Transactions on Control Systems Technology 2015;23:1685–700.

100. Transition Ecologique M de la. Chiffres clés des énergies renouvelables. https://www.statistiques.developpement-durable.gouv.fr/edition-numerique/chiffres-cles-energies-renouvelables-2021/1-les-energies-renouvelables-en-france. Accessed: 2024-01-09. 2021.

101. Trindade RS, d'Ambrosio C, Frangioni A, and Gentile C. Comparing perspective reformulations for piecewise-convex optimization. Operations Research Letters 2023;51:702–8.

102. Turner L. Variants of the shortest path problem. Algorithmic Operations Research 2011;6:91–104.

103. Ulungu EL and Teghem J. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. Foundations of computing and decision sciences 1995;20:149–65.

104. Vielma JP and Nemhauser GL. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Mathematical Programming 2011;128:49–72.

105. Vigerske S and Gleixner A. SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. Optimization Methods and Software 2018;33:563–93.

106. Visée M, Teghem J, Pirlot M, and Ulungu EL. Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem. Journal of Global Optimization 1998;12:139–55.

107. Wolsey LA. Valid inequalities for 0–1 knapsacks and MIPs with generalised upper bound constraints. Discrete Applied Mathematics 1990;29:251–61.

108. Yu G and Yang J. On the Robust Shortest Path Problem. Computers & Operations Research 1998;25:457–68.

109. Zhu X and Wilhelm WE. Three-stage approaches for optimizing some variations of the resource constrained shortest-path sub-problem in a column generation context. European journal of operational research 2007;183:564–77.

110. Zhu X and Wilhelm WE. A three-stage approach for the resource-constrained shortest path as a sub-problem in column generation. Computers & Operations Research 2012;39:164–78.

111. Zukerman M, Jia L, Neame T, and Woeginger GJ. A polynomially solvable special case of the unbounded knapsack problem. Operations Research Letters 2001;29:13–6.

Appendix for the model comparison

Table of contents

A.1 Solver description

ANTIGONE [73] is based on an sBB algorithms. The problem is reformulated in order to find special structures. Once the structures are found, the relaxation of the problem is solved. The search space is

split and the process repeated until convergence of the upper and the lower bounds. Upper bounds are computed with local optimization algorithms. Only twice differentiable functions, that are not trigonometrical functions, are supported by ANTIGONE.

BARON [98] implements a deterministic Branch and Reduce algorithm. This algorithm contains constraint programming, interval analysis and duality techniques for tightening variables bounds. Heuristics, cutting planes and parallelism are combined with the Branch and Reduce algorithm. Trigonometrical functions and **max** functions are not supported.

COUENNE [12] implements an sBB with linearization, bound reductions and branching method. The main four components are: reformulation, separation of linearization cuts, branching rules and bound tightening methods. COUENNE only supports functions that can be reformulated into univariate functions and does not support function **max**.

LINDOGlobal [68] is the only solver that does not directly implements an sBB algorithm. Instead, it implements a branch and cut algorithm that breaks the model into sub-problems. The sub-problems are further split until each sub-problem is convex. The sub-problems are then solved with a BB or sBB algorithm. LINDOGlobal supports most non-linearities, and binary operators such as AND, OR and NOT.

SCIP [105] implements an sBB, where the non-linearities are represented within graphs. These graphs help finding convex non-linearities, and reformulating the non-linear functions. During the solving process, SCIP also adds various cuts, depending on the non-linearities. Bound tightening methods are also applied. Trigonometrical functions are not supported by SCIP and it is the only solver which requires a linear objective function.

CPLEX [26] implements a quite effective multipurpose Branch and Cut algorithm, which generates automatically various cuts [26]. Furthermore it is paired with pre-processing and heuristics.

A.2 Five parameters logistic function

A ***5PL*** is the following function, where x is a variable and y_1 to y_5 the parameters:

$$5PL(x, y_1, y_2, y_3, y_4, y_5) = y_4 + \frac{-y_4}{(1 + (\frac{x-y_1}{y_3})^{y_2})^{y_5}}$$

In the context of the 1-HUC^{NL} problem, variable x is the water-flow d_t. The ***5PL*** has a shape similar to a more common function, the sigmoid:

$$sig(x, y_1', y_2', y_3') = \frac{y_3'}{1 + e^{-y_2'(x-y_1')}}$$

The advantages of the ***5PL*** is that it is more flexible than a sigmoid. The sigmoid is necessarily symmetric with respect to its inflection point, whereas ***5PL*** is not. However, a ***5PL*** function is not defined if $x < y_1$, which can occur when representing a unit by a ***5PL*** function. To adapt the ***5PL*** function to the use case

of the 1-HUC^{NL} problem, it is possible to insert a **max** function inside the ***5PL*** function as follows:

$$5PL(x, y_1, y_2, y_3, y_4, y_5) = y_4 + \frac{-y_4}{\left(1 + \left(\frac{\mathbf{max}(0, x - y_1)}{y_3}\right)^{y_2}\right)^{y_5}}$$

With this modification, if $x < y_1$ then the ***5PL*** function is equal to $y_4 + (-y_4/1) = 0$, if $x \geq y_1$, the ***5PL*** has the same behaviour as previously defined.

A.3 Instance description

The instances are derived from the following parameter sets A and B, by changing the value of only one parameter at a time. The idea is to evaluate the impact of the parameters on the resolution and the solution with multiple metrics. **Table A.1** shows the parameters of each parameter set.

Table A.1: Parameter sets

Parameter set A	Parameter set B
$V_0^1 = 500$, $V_0^2 = 200$	$V_0^1 = 90$, $V_0^2 = 10$
$T = 4$	$T = 4$
$\overline{V}_t^1 = 1000$, $\underline{V}_t^1 = 0 \ \forall t \leq T$	$\overline{V}_t^1 = 100$, $\underline{V}_t^1 = 0 \ \forall t \leq T$
$\overline{V}_t^2 = 500$, $\underline{V}_t^2 = 0 \ \forall t \leq T$	$\overline{V}_t^2 = 90$, $\underline{V}_t^2 = 0 \ \forall t \leq T$
$\underline{D} = 0$, $\overline{D} = 25$	$\underline{D} = 0$, $\overline{D} = 8$
$\underline{P}_t = 0$, $\overline{P}_t = 15 \ \forall t \leq T$	$\underline{P}_t = 0$, $\overline{P}_t = 32 \ \forall t \leq T$
$\Phi^1 = 230$, $\Phi^2 = 0$	$\Phi^1 = 850$, $\Phi^2 = 0$
$\Lambda = [0.2, 0.15, 0.1, 0.2]$	$\Lambda = [0.1, 0.2, 0.5, 0.4]$
$A_t^1 = A_t^2 = 0 \ \forall t \leq T$	$A_t^1 = A_t^2 = 0 \ \forall t \leq T$
$\gamma = [0, 0.1, 5, 0.7]$	$\gamma = [100, 0.2, 2, 0.6]$

The modified features are the following:

- Size of the instance;
- Equality constraints;
- Number of inflection points of the non-linear function;
- Degree of non-linearity of the function;
- Sensitivity of the decision variables to the non-linear effect.

In these parameter sets, the maximum and minimum volumes are artificially large, to see the impact of each feature. Below, we justify the choice for each feature and explain how the changes are instantiated on the 1-HUC^{NL} problem. Note that every instance is built such that there is at least one feasible solution with continuous water flows d_t.

A.3.1 Size of the instance

Larger instances are in general harder to solve as they contain more variables and constraints. In the case of the 1-HUC^{NL} problem, larger instances considered will have a larger number of time periods T. Increasing T exponentially increases the number of feasible solutions. Three instances are considered, A-T-1 to A-T-3 (resp. B-T-1 to B-T-3) corresponding to the variations of the parameter set A (resp. B) with 4, 7 and 10 time periods T. To take into account more time periods, prices are supplemented as follows: $\Lambda = [0.2, 0.15, 0.1, 0.2, 0.1, 0.05, 0.1, 0.2, 0.15, 0.05]$ (resp. $\Lambda = [0.1, 0.2, 0.5, 0.4, 0.3, 0.2, 0.3, 0.5, 0.4, 0.2]$). These instances are such that the volume of each reservoir can not reach the maximum or minimum volume. The water flow will not be affected by the bounds on the volume, in contrary to some other sets of instances.

A.3.2 Equality constraints

Equality constraints can highly affect the resolution. Indeed, equality constraints drastically reduce the number of feasible solutions and can also be hard to satisfy. Moreover, depending on the approximation used in the model, equality constraints may lead to non efficient solutions. In the case of the 1-HUC^{NL} problem, target volumes are equality constraints, when $\underline{V}_t^1 = \overline{V}_t^1$ for a time period t. Six instances are considered, A-E-1 to A-E-6 (resp. B-E-1 to B-E-6) which are variations of parameter set A (resp. B), where target volumes are only for the last time period T. For A-E-1 to A-E-3 (resp. B-E-1 to B-E-3) the target volumes are 480, 450 and 420 (resp. 80, 70 and 60). For A-E-4 to A-E-6 (resp B-E-4 to B-E-6), the target volumes are 500 (resp. 90), but the additional intake of water at the last time period are 20, 50 and 80 (resp. 10, 20, 30). One can notice that for instance A-E-1, the difference between the initial and the target volume is 20, while for instance A-E-4 it is 0, but the additional intake of water is 20. Thus, feasible solutions for A-E-1 are feasible solutions for A-E-4 and vice-versa. Instances A-E-2 and A-E-5, B-E-1 and B-E-4 and so on are built similarly.

A.3.3 Number of inflection points of the non-linear function

With a different number of inflection points, the shape of a non-linear function is changed, which can lead to more local optimal solutions, or less efficient under-estimators. The functions used to under-estimate and approximate the functions are also changed. Thus, the resolution and approximation error could be impacted by the number of inflection points of the non-linear function. In the case of the 1-HUC^{NL} problem, the number of inflection points of the power function can be changed by defining a larger number of smaller units. We still have the same maximum power and maximum water flow, only the shape of the power function is different. Three instances are considered, A-N-1 to A-N-3 (resp. B-N-1 to B-N-3) which are variations of the parameter set A (resp. B) with 2, 4 and 6 units.

A.3.4 Degree of non-linearity of the non-linear function

As for the number of inflection points, changing the degree of non-linearity is another way to change the shape of a function. Thus, the resolution and approximation error could be affected by the degree of non-linearity of the functions. In the case of the 1-HUC^{NL} problem, one way to increase the non-linearity of the power function is to change the water flow when each unit starts and stops, increasing or reducing the degree of curvature for each concave part of the function, as represented by. The resulting power function for the plant can be quasi-linear or have a high degree of non-linearity. In addition, it also changes the domain of some variables. Six instances are considered, A-D-1 to A-D-6 (resp. B-D-1 to B-D-6) being variations of parameter set A (resp. B). Instances A-D-1 and A-D-4 feature a quasi linear function, with $\overline{D} = 22$ and $\overline{P}_t = 14.5$, $\forall t \leq T$, instances A-D-2 and A-D-5 correspond to a non-linear function, with $\overline{D} = 25$ and $\overline{P}_t = 15$, and instances A-D-3 and A-D-6 use a very non-linear function, with $\overline{D} = 28$ and $\overline{P}_t = 16$. The target volume for instances A-D-4 to A-D-6 is 460. Similarly, instances B-D-1 and B-D-4 feature a quasi linear function, with $\overline{D} = 6$ and $\overline{P}_t = 28$, instances B-D-2 and B-D-5 feature a non-linear function, with $\overline{D} = 8$ and $\overline{P}_t = 32$, and instances B-D-3 and B-D-6 feature a very non-linear function, with $\overline{D} = 10$ and $\overline{P}_t = 34$. The target volume for instances B-D-4 to B-D-6 is 75.

A.3.5 Sensitivity of the decision variables to the non-linear effect

Depending on the problem, decision variables can have a very large, or very small impact on the non-linearities. When the impact is small, it is possible that some simplifications of the problem would not induce large approximation errors. In the case of the 1-HUC^{NL} problem, the sensitivity of the decision variables to the non-linear effect can change by considering larger or smaller reservoirs. Two instances are considered, A-S-1 and A-S-2 (resp. B-S-1 and B-S-2) are variations of parameter set A (resp. B). Instance A-S-2 is similar to A-S-1, but has all initial, maximal and minimal volumes multiplied by 100, and supplemented prices $\Lambda = [0.005, 0.00375, 0.0025, 0.005]$. Analogously, B-S-2 is similar to B-S-1 with initial, maximal and minimal volumes all multiplied by 100, and adapted prices $\Lambda = [0.005, 0.01, 0.025, 0.02]$. The unitary prices Λ are reduced in order to obtain similar solutions for instances A-S-1 and A-S-2 (resp. B-S-1 and B-S-2).

One can compute bounds on the variation of the volume, by calculating the maximum and minimum water processed while respecting the capacities. These bounds give an interval for the final volume in the reservoirs. It is then possible to compute the maximum difference in terms of volume between two feasible solutions, and compare it to the capacity of the reservoirs in order to predict if the instance might induce high volume variations or not. The sensitivity S can be computed as follows:

$$S = \frac{\overline{D} \cdot T - \underline{D} \cdot T}{\min(\overline{V}_T^1 - \underline{V}_T^1, \overline{V}_T^2 - \underline{V}_T^2)}$$

For instance A-S-1, the sensitivity is 100/500 = 0.2, for instance A-S-2: 0.002, for instance B-S-1: 0.36 and for instance B-S-2: 0.0036. Note that the parameter set A (resp. B) has the same sensitivity as

instance A-S-1 (resp. B-S-1).

Table 9 summarizes these instances and their features.

Table A.2: Instance features

Instances	features	Modified parameter
A-T-1 to A-T-3 B-T-1 to B-T-3	Size of the instance	Number of time periods
A-E-1 to A-E-6 B-E-1 to B-E-6	Equality constraints	Different target volumes, with and without additional intakes of water
A-N-1 to A-N-3 B-N-1 to B-N-3	Number of inflection points	Number of units
A-D-1 to A-D-6 B-D-1 to B-D-6	Degree of non-linearity	Quasi-linear, non-linear or very linear function, with and without target volumes
A-S-1, A-S-2, B-S-1, B-S-2	Sensitivity of the decision variables	Size of the reservoirs

A.4 Analysis of the impact of the instance features

Let us analyse the impact of each feature of a 1-HUC^{NL} problem instance on the resolution. The tables related to the results described in the following section are in **Appendix A.5**.

A.4.1 Size of the instance

Changing the number of time periods (instances A-T-1 to A-T-3 and B-T-1 to B-T-3) has a big effect on the resolution. Indeed, we see from **Table A.3** and **Table A.4** that configurations with more time periods require a drastically increased CT compared to configurations with fewer time periods. The most salient case is for the 1-HUC^{NL} problem where with $T = 10$ the only configurations solved under three hours by their VBS are with one of the following four models: $M_{2D\text{-}poly}$, M_{bilin}, $M_{PWL\text{-}1}$ and $M_{PWL\text{-}2}$. Moreover, with $T = 7$ configurations with model $M_{5PL\text{-}max}$ are never solved by their VBS within three hours. The AE also increases for configurations with instances with more time periods. It is especially visible for the fixed-head 1-HUC^{NL} problem, with $T = 10$ the minimal AE is around 20% using models $M_{5PL\text{-}max}$, and the average AE are around 40% at least.

As there are more time periods, more variables and constraints are introduced, exponentially increasing the number of feasible solutions. The reason why the AE increases is due to the fact that errors are propagated through the time periods. Also, for the fixed-head 1-HUC^{NL} problem, more time periods mean, in general, more water processed. The volume varies with a higher magnitude from the initial volume when there are more time periods, leading to larger AE when considering a fixed head.

A.4.2 Equality constraints

Taking fixed target volumes (instances A-E-1 to A-E-6 , B-E-1 to B-E-6, A-D-1 to A-D-6 and B-D-1 to B-D-6) has a non-homogeneous impact on the resolution. From **Table A.5** and **Table A.6** we notice that configurations with target volumes reduce the CT required for the 1-HUC^{NL} problem, compared to configurations without target volumes. For the AE, we notice multiple behaviours. For most models, configurations with target volumes yields to smaller average AE, but higher maximal AE, compared to configurations without target volumes. Non-represented results also showed that ANTIGONE solves less than 25% of configurations with target volumes whereas it solves more than 60% of configurations without target volumes. A similar but less marked behaviour is noticed for COUENNE.

The decreased CT is probably due to the fact that fewer solutions are feasible. The reduced AE are due to the target volume being very close to the initial volume for some instances. Less volume is processed, meaning a smaller power, and smaller AE. Besides, for the fixed-head 1-HUC^{NL} problem, it also means less errors due to the fixed head, as the volume may not vary to much from the initial volume. We notice that some configurations with model M_{op} are not solved, for both the 1-HUC^{NL} problem and the fixed-head 1-HUC^{NL} problem. This is because the target volume may not be reachable with the finite set of water flows.

A.4.3 Degree of non-linearity

Changing the non-linearity of the power function (instances A-D-1 to A-D-6 and B-D-1 to B-D-6) can have an impact on the CT and the AE in the case of the 1-HUC^{NL} problem, but only on the AE for the fixed-head 1-HUC^{NL} problem. From **Table A.7** and **Table A.8**, we notice that configurations with pronounced non-linearities have larger CT for the 1-HUC^{NL} problem than configurations with quasi-linear functions. The configurations with non-linear models also have larger AE with pronounced non-linear functions, for both the 1-HUC^{NL} problem and the fixed-head 1-HUC^{NL} problem. The AE for configurations with linear model is not affected. We also see that all the configurations with model $M_{2D\text{-}poly}$ are infeasible with every solver when the instance has a pronounced non-linear function, even if there exist feasible solutions for the instance.

The general increase of the AE for non-linear models can be explained by two reasons. Firstly, by instance construction, the units are the same for every instances, and the water-flow interval for each unit is changed in order to have a different degree of non-linearity. As such, it is possible that fewer non-linear functions can closely approximate the function of M_{ref} on a larger interval. Secondly, a highly non-linear function can be harder to approximate by simpler functions, leading to larger AE for every model.

A.4.4 Number of inflection points

Changing the number of units (instances A-N-1 to A-N-3 and B-N-1 to B-N-3) has only a noticeable impact on models representing each unit explicitly, namely $M_{5PL\text{-}max}$, $M_{5PL\text{-}bin}$ and $M_{2D\text{-}poly}$. Indeed, **Table A.9** and **Table A.10** show the increased CT required for configurations with these models and with instances with more units. Also, increased number of units reduces the degree of non-linearity. Thus it is possible to see similar behaviours as when changing the degree of non-linearity.

The reason why the CT increases for configurations with one of the four mentioned models and an instance with many units is because as they represent each unit explicitly, more variables and constraints are required.

A.4.5 Sensitivity of the decision variables to the non-linear effect

In order to have negligible variation of the volume, the volumes can be set to larger values than the water flows. **Table A.11** and **Table A.12** show that the CT tends to be smaller for configurations with large volumes compared to configurations with smaller volumes. Larger volumes usually lead to an improvement of the AE for the fixed-head 1-HUC^{NL} problem. However, the maximal AE of PWL models can be very large with large volumes. More precisely, with a PWL models, half the configurations has a large AE, and the other half has a smaller AE, compared to configurations with small volumes

The improvement of AE for the fixed-head 1-HUC^{NL} problem is because with small variations, the volume is very similar to the initial volume at any time period. The AE from the fixed-head becomes very small. The high AE of the PWL models can be explained as follows. These models only consider a family of univariate PWL function for a finite set of possible volumes. It is then possible that the volume is never similar to the volumes used by this family of functions.

A.5 Numerical experiments when partitioning instances

- %S: proportion of configuration solved;
- min-CT, max-CT, avg-CT: minimum, maximum, and average CT for every solved configurations;
- min-AE, max-AE, avg-AE: minimum, maximum and average AE for every solved configurations.

A.5.1 Size of the instance

Table A.3 and **Table A.4** represents the proportion of configurations with instances with 4, 7 and 10 time periods and each model solved by their VBS, and related minimum, maximum and average CT and AE.

Table A.3: Proportion of configurations solved with their VBS, CT and AE statistics for the 1-HUC^{NL} problem for different number of time periods (instances A-T-1 to A-T-3 and B-T-1 to B-T-3)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
T =4	$M_{5PL\text{-}max}$	100.0	103.34	186.38	144.86	0.2	0.4	0.3
	$M_{5PL\text{-}bin}$	100.0	15.28	28.47	21.88	0.2	0.4	0.3
	$M_{2D\text{-}poly}$	100.0	0.43	0.69	0.56	0.8	3.9	2.4
	M_{op}	100.0	1.47	2.25	1.86	0.3	0.4	0.3
	M_{bilin}	100.0	0.03	0.05	0.04	24.3	26.1	25.2
	$M_{PWL\text{-}3}$	100.0	0.4	7.06	3.73	1.3	3.5	2.4
	$M_{PWL\text{-}2}$	100.0	0.09	0.18	0.14	5.8	6.5	6.2
	$M_{PWL\text{-}1}$	100.0	0.01	0.02	0.01	11.8	67.9	39.9
T =7	$M_{5PL\text{-}max}$	0	-	-	-	-	-	-
	$M_{5PL\text{-}bin}$	100.0	10 367.97	10 367.97	10 367.97	0.4	0.4	0.4
	$M_{2D\text{-}poly}$	100.0	8.31	36.18	22.25	0.8	3.3	2.0
	M_{op}	100.0	78.69	574.13	326.41	4.0	14.3	9.2
	M_{bilin}	100.0	0.06	0.26	0.16	24.1	31.3	27.7
	$M_{PWL\text{-}3}$	100.0	39.12	39.12	39.12	1.3	1.3	1.3
	$M_{PWL\text{-}2}$	100.0	0.15	2.88	1.51	5.7	8.7	7.2
	$M_{PWL\text{-}1}$	100.0	0.03	0.06	0.04	27.9	67.9	47.9
T =10	$M_{5PL\text{-}max}$	0	-	-	-	-	-	-
	$M_{5PL\text{-}bin}$	0	-	-	-	-	-	-
	$M_{2D\text{-}poly}$	100.0	44.71	344.08	194.39	0.8	14.1	7.5
	M_{op}	100.0	7062.75	7062.75	7062.75	78.6	78.6	78.6
	M_{bilin}	100.0	0.07	0.09	0.08	40.4	46.3	43.3
	$M_{PWL\text{-}3}$	50.0	558.73	558.73	558.73	1.1	1.1	1.1
	$M_{PWL\text{-}2}$	100.0	0.46	19.05	9.76	7.4	14.0	10.7
	$M_{PWL\text{-}1}$	100.0	0.05	0.16	0.11	32.4	66.8	49.6

A.5.2 Equality constraints

Table A.5 and **Table A.6** represents the proportion of configurations with instances with and without target volumes and each model solved by their VBS, and related minimum, maximum and average CT and AE.

A.5.3 Degree of non-linearity

Table A.7 and **Table A.8** represents the proportion of configurations with instances with a quasi linear, a non-linear and a very non-linear function and each model solved by their VBS, and related minimum, maximum and average CT and AE.

Table A.4: Proportion of configurations solved with their VBS, CT and AE statistics for the fixed-head 1-HUC^{NL} problem for different number of time periods (instances A-T-1 to A-T-3 and B-T-1 to B-T-3)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
T =4	$M_{5PL\text{-}max}$	100.0	0.11	0.12	0.11	10.7	20.6	15.7
	$M_{5PL\text{-}bin}$	100.0	0.25	0.4	0.33	10.7	20.6	15.7
	$M_{2D\text{-}poly}$	100.0	0.06	0.08	0.07	14.6	27.3	20.9
	M_{op}	100.0	0.0	0.01	0.01	10.3	21.0	15.7
	M_{bilin}	100.0	0.0	0	0.0	41.0	66.6	53.8
	$M_{PWL\text{-}3}$	100.0	0.02	0.02	0.02	20.2	21.9	21.0
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	19.9	21.8	20.9
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	21.9	22.2	22.0
T =7	$M_{5PL\text{-}max}$	100.0	0.11	0.14	0.12	12.4	32.1	22.2
	$M_{5PL\text{-}bin}$	100.0	2.77	10.14	6.46	18.6	32.1	25.4
	$M_{2D\text{-}poly}$	100.0	0.06	0.1	0.08	23.6	43.4	33.5
	M_{op}	100.0	0.01	0.01	0.01	17.8	33.1	25.5
	M_{bilin}	100.0	0.0	0	0.0	60.2	115.6	87.9
	$M_{PWL\text{-}3}$	100.0	0.03	0.03	0.03	31.5	32.1	31.8
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	30.8	31.6	31.2
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	30.2	35.7	33.0
T =10	$M_{5PL\text{-}max}$	100.0	0.11	0.15	0.13	19.6	59.2	39.4
	$M_{5PL\text{-}bin}$	100.0	6.25	50.0	28.12	29.7	59.2	44.5
	$M_{2D\text{-}poly}$	100.0	0.08	0.08	0.08	34.7	67.3	51.0
	M_{op}	100.0	0.01	0.01	0.01	28.4	49.0	38.7
	M_{bilin}	100.0	0.0	0	0.0	78.7	149.9	114.3
	$M_{PWL\text{-}3}$	100.0	0.04	0.04	0.04	44.7	47.0	45.9
	$M_{PWL\text{-}2}$	100.0	0.01	0.02	0.01	44.2	46.1	45.2
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	42.9	52.8	47.8

A.5.4 Number of inflection points

Table A.9 and **Table A.10** represents the proportion of configurations with instances with 2, 6 and 6 units and each model solved by their VBS, and related minimum, maximum and average CT and AE.

A.5.5 Sensitivity of the decision variables to the non-linear effect

Table A.11 and **Table A.12** represents the proportion of configurations with instances with small and large volumes and each model solved by their VBS, and related minimum, maximum and average CT and AE.

Table A.5: Proportion of configurations solved with their VBS, CT and AE statistics for the 1-HUC^{NL} problem with and without target volumes (instances A-T-1, B-T-1, A-E-1 to A-E-6, B-E-1 to B-E-6, A-D-1 to A-D-6 and B-D-1 to B-D-6)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
No target volume	$M_{5PL\text{-}max}$	100.0	51.99	212.62	148.12	0.0	12.3	1.9
	$M_{5PL\text{-}bin}$	100.0	4.08	31.84	21.97	0.0	12.3	1.9
	$M_{2D\text{-}poly}$	100.0	0.4	0.81	0.53	0.0	7.2	2.8
	M_{op}	100.0	1.47	2.95	1.94	0.1	12.7	1.9
	M_{bilin}	100.0	0.03	0.06	0.04	19.4	32.3	26.2
	$M_{PWL\text{-}3}$	100.0	0.4	8.98	3.6	0.9	4.2	2.5
	$M_{PWL\text{-}2}$	100.0	0.09	0.18	0.13	4.4	6.5	5.8
	$M_{PWL\text{-}1}$	100.0	0.01	0.03	0.02	10.0	70.6	39.8
target volume	$M_{5PL\text{-}max}$	100.0	0.66	399.96	94.73	0.0	9.3	1.1
	$M_{5PL\text{-}bin}$	100.0	0.1	18.31	7.66	0.0	9.3	0.8
	$M_{2D\text{-}poly}$	100.0	0.09	0.48	0.29	0.2	4.2	1.4
	M_{op}	100.0	0.56	23.96	3.81	0.0	10.5	1.2
	M_{bilin}	100.0	0.03	0.13	0.07	0.5	27.4	12.4
	$M_{PWL\text{-}3}$	100.0	0.14	1.0	0.44	0.8	22.9	3.4
	$M_{PWL\text{-}2}$	100.0	0.01	0.17	0.07	1.3	30.8	6.6
	$M_{PWL\text{-}1}$	100.0	0.01	0.02	0.01	4.9	69.4	33.0

Table A.6: Proportion of configurations solved with their VBS, CT and AE statistics for the fixed-head 1-HUC^{NL} problem with and without target volumes (instances A-T-1, B-T-1, A-E-1 to A-E-6, B-E-1 to B-E-6, A-D-1 to A-D-6 and B-D-1 to B-D-6)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
No target volume	$M_{5PL\text{-}max}$	100.0	0.1	0.12	0.11	7.3	24.1	15.9
	$M_{5PL\text{-}bin}$	100.0	0.2	0.48	0.35	8.8	28.7	17.8
	$M_{2D\text{-}poly}$	100.0	0.06	0.14	0.08	9.6	28.4	20.8
	M_{op}	100.0	0.0	0.01	0.01	8.0	27.8	17.4
	M_{bilin}	100.0	0.0	0	0.0	29.0	87.5	56.5
	$M_{PWL\text{-}3}$	100.0	0.02	0.03	0.03	17.7	27.5	21.4
	$M_{PWL\text{-}2}$	100.0	0.01	0.02	0.01	17.2	27.4	21.1
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	18.4	29.2	22.2
target volume	$M_{5PL\text{-}max}$	100.0	0.1	0.93	0.2	2.0	19.5	7.6
	$M_{5PL\text{-}bin}$	100.0	0.03	1.06	0.52	1.0	80.0	14.0
	$M_{2D\text{-}poly}$	100.0	0.06	0.23	0.13	0.6	28.8	10.4
	M_{op}	100.0	0.0	0.02	0.01	1.0	19.9	7.1
	M_{bilin}	100.0	0.0	0	0.0	2.7	49.8	21.2
	$M_{PWL\text{-}3}$	100.0	0.02	0.17	0.07	0.8	33.5	10.6
	$M_{PWL\text{-}2}$	100.0	0.01	0.04	0.02	0.8	33.5	10.6
	$M_{PWL\text{-}1}$	100.0	0.0	0.01	0.01	0.9	33.6	10.8

Table A.7: Proportion of configurations solved with their VBS, CT and AE statistics for the 1-HUC^{NL} problem for different degree of non-linearity (instances A-D-1 to A-D-6 and B-D-1 to B-D-6)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
Quasi linear	$M_{5PL\text{-}max}$	100.0	2.41	399.96	162.15	0.0	0.5	0.3
	$M_{5PL\text{-}bin}$	100.0	3.18	31.84	13.63	0.0	0.5	0.3
	$M_{2D\text{-}poly}$	100.0	0.23	0.4	0.35	0.0	7.2	2.9
	M_{op}	100.0	1.43	1.68	1.54	0.1	0.5	0.2
	M_{bilin}	100.0	0.03	0.08	0.05	10.2	27.7	18.1
	$M_{PWL\text{-}3}$	100.0	0.2	3.39	1.15	1.0	3.6	2.2
	$M_{PWL\text{-}2}$	100.0	0.1	0.14	0.12	4.4	6.2	5.4
	$M_{PWL\text{-}1}$	100.0	0.01	0.02	0.02	10.0	68.7	38.8
Non-linear	$M_{5PL\text{-}max}$	100.0	43.21	239.32	148.84	0.2	0.4	0.3
	$M_{5PL\text{-}bin}$	100.0	8.83	29.02	17.6	0.2	0.4	0.3
	$M_{2D\text{-}poly}$	100.0	0.21	0.81	0.46	0.8	3.9	1.9
	M_{op}	100.0	1.08	2.93	1.98	0.3	0.4	0.3
	M_{bilin}	100.0	0.04	0.08	0.05	0.9	26.1	16.7
	$M_{PWL\text{-}3}$	100.0	0.15	6.73	2.01	1.3	3.6	2.6
	$M_{PWL\text{-}2}$	100.0	0.03	0.16	0.1	5.8	7.1	6.4
	$M_{PWL\text{-}1}$	100.0	0.01	0.02	0.02	11.8	68.4	40.0
Very non-linear	$M_{5PL\text{-}max}$	100.0	62.31	208.98	151.88	1.0	12.3	6.2
	$M_{5PL\text{-}bin}$	100.0	0.56	27.81	15.87	0.0	12.3	5.7
	$M_{2D\text{-}poly}$	0	-	-	-	-	-	-
	M_{op}	100.0	0.83	3.58	2.29	0.9	12.7	6.3
	M_{bilin}	100.0	0.03	0.1	0.06	7.8	32.3	21.7
	$M_{PWL\text{-}3}$	100.0	0.37	8.98	2.87	0.9	4.2	2.6
	$M_{PWL\text{-}2}$	100.0	0.09	0.17	0.14	5.2	7.2	6.3
	$M_{PWL\text{-}1}$	100.0	0.02	0.03	0.02	10.0	70.6	40.0

Table A.8: Proportion of configurations solved with their VBS, CT and AE statistics for the fixed-head 1-HUC^{NL} problem for different degree of non-linearity (instances A-D-1 to A-D-6 and B-D-1 to B-D-6)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
Quasi linear	$M_{5PL\text{-}max}$	100.0	0.1	0.13	0.11	7.1	18.4	10.2
	$M_{5PL\text{-}bin}$	100.0	0.2	0.82	0.46	7.1	18.4	10.5
	$M_{2D\text{-}poly}$	100.0	0.06	0.14	0.1	6.8	28.4	14.3
	M_{op}	100.0	0.01	0.01	0.01	7.9	16.8	10.9
	M_{bilin}	100.0	0.0	0	0.0	21.1	53.2	31.4
	$M_{PWL\text{-}3}$	100.0	0.02	0.03	0.03	7.5	18.5	14.8
	$M_{PWL\text{-}2}$	100.0	0.01	0.02	0.01	7.4	18.4	14.7
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	6.8	19.8	15.2
Non-linear	$M_{5PL\text{-}max}$	100.0	0.11	0.17	0.12	6.2	20.6	11.7
	$M_{5PL\text{-}bin}$	100.0	0.32	1.03	0.56	6.2	20.6	11.5
	$M_{2D\text{-}poly}$	100.0	0.06	0.23	0.15	7.5	27.3	14.1
	M_{op}	100.0	0.0	0.01	0.01	6.1	21.0	11.5
	M_{bilin}	100.0	0.0	0	0.0	18.7	66.6	37.3
	$M_{PWL\text{-}3}$	100.0	0.02	0.03	0.03	7.8	21.9	16.1
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	7.6	21.8	16.0
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	6.1	22.2	15.9
Very non-linear	$M_{5PL\text{-}max}$	100.0	0.1	0.12	0.11	10.1	24.1	15.9
	$M_{5PL\text{-}bin}$	100.0	0.35	1.01	0.56	10.3	28.7	19.9
	$M_{2D\text{-}poly}$	100.0	0.06	0.19	0.11	6.8	26.8	15.7
	M_{op}	100.0	0.01	0.02	0.01	5.0	27.8	18.7
	M_{bilin}	100.0	0.0	0	0.0	26.1	87.5	52.6
	$M_{PWL\text{-}3}$	100.0	0.03	0.12	0.05	8.1	27.5	18.8
	$M_{PWL\text{-}2}$	100.0	0.01	0.03	0.02	8.0	27.4	18.5
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	12.2	29.2	20.2

Table A.9: Proportion of configurations solved with their VBS, CT and AE statistics for the 1-HUC^{NL} problem for different number of units (instances A-N-1 to A-N-3 and B-N-1 to B-N-3)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
K =2	$M_{5PL\text{-}max}$	100.0	103.34	186.38	144.86	0.2	0.4	0.3
	$M_{5PL\text{-}bin}$	100.0	15.28	28.47	21.88	0.2	0.4	0.3
	$M_{2D\text{-}poly}$	100.0	0.43	0.69	0.56	0.8	3.9	2.4
	M_{op}	100.0	1.47	2.25	1.86	0.3	0.4	0.3
	M_{bilin}	100.0	0.03	0.05	0.04	24.3	26.1	25.2
	$M_{PWL\text{-}3}$	100.0	0.4	7.06	3.73	1.3	3.5	2.4
	$M_{PWL\text{-}2}$	100.0	0.09	0.18	0.14	5.8	6.5	6.2
	$M_{PWL\text{-}1}$	100.0	0.01	0.02	0.01	11.8	67.9	39.9
K =4	$M_{5PL\text{-}max}$	100.0	1764.35	5988.44	3876.39	0.1	0.4	0.2
	$M_{5PL\text{-}bin}$	100.0	13.75	63.73	38.74	0.1	0.4	0.2
	$M_{2D\text{-}poly}$	100.0	2.76	3.47	3.12	0.4	3.1	1.8
	M_{op}	100.0	0.49	1.57	1.03	0.0	0.3	0.1
	M_{bilin}	100.0	0.04	0.05	0.04	19.6	22.1	20.9
	$M_{PWL\text{-}3}$	100.0	0.75	5.73	3.24	0.3	3.3	1.8
	$M_{PWL\text{-}2}$	100.0	0.11	0.15	0.13	4.0	6.4	5.2
	$M_{PWL\text{-}1}$	100.0	0.02	0.02	0.02	12.0	70.9	41.5
K =6	$M_{5PL\text{-}max}$	100.0	1055.49	1055.49	1055.49	0.0	0	0.0
	$M_{5PL\text{-}bin}$	100.0	7.63	36.01	21.82	0.0	0.3	0.1
	$M_{2D\text{-}poly}$	100.0	1.11	3.81	2.46	0.7	8.7	4.7
	M_{op}	100.0	0.38	1.3	0.84	0.0	0.9	0.5
	M_{bilin}	100.0	0.04	0.04	0.04	13.6	20.4	17.0
	$M_{PWL\text{-}3}$	100.0	0.5	6.63	3.56	0.4	4.2	2.3
	$M_{PWL\text{-}2}$	100.0	0.14	0.22	0.18	3.7	6.4	5.1
	$M_{PWL\text{-}1}$	100.0	0.02	0.02	0.02	10.6	75.8	43.2

Table A.10: Proportion of configurations solved with their VBS, CT and AE statistics for the fixed-head 1-HUC^{NL} problem for different number of units (instances A-N-1 to A-N-3 and B-N-1 to B-N-3)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
K =2	$M_{5PL\text{-}max}$	100.0	0.11	0.12	0.11	10.7	20.6	15.7
	$M_{5PL\text{-}bin}$	100.0	0.25	0.4	0.33	10.7	20.6	15.7
	$M_{2D\text{-}poly}$	100.0	0.06	0.08	0.07	14.6	27.3	20.9
	M_{op}	100.0	0.0	0.01	0.01	10.3	21.0	15.7
	M_{bilin}	100.0	0.0	0	0.0	41.0	66.6	53.8
	$M_{PWL\text{-}3}$	100.0	0.02	0.02	0.02	20.2	21.9	21.0
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	19.9	21.8	20.9
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	21.9	22.2	22.0
K =4	$M_{5PL\text{-}max}$	100.0	0.11	0.33	0.22	12.2	24.9	18.5
	$M_{5PL\text{-}bin}$	100.0	0.14	0.76	0.45	14.4	22.4	18.4
	$M_{2D\text{-}poly}$	100.0	0.06	0.09	0.07	10.3	11.2	10.8
	M_{op}	100.0	0.01	0.01	0.01	11.6	22.0	16.8
	M_{bilin}	100.0	0.0	0	0.0	36.7	56.7	46.7
	$M_{PWL\text{-}3}$	100.0	0.02	0.03	0.03	22.7	23.4	23.0
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	22.8	23.0	22.9
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	21.7	26.5	24.1
K =6	$M_{5PL\text{-}max}$	100.0	0.1	0.33	0.22	13.0	25.4	19.2
	$M_{5PL\text{-}bin}$	100.0	0.03	0.82	0.42	24.7	90.2	57.5
	$M_{2D\text{-}poly}$	100.0	0.06	0.07	0.07	13.9	15.6	14.8
	M_{op}	100.0	0.0	0	0.0	10.3	24.0	17.1
	M_{bilin}	100.0	0.0	0	0.0	35.0	48.4	41.7
	$M_{PWL\text{-}3}$	100.0	0.02	0.03	0.03	23.1	25.1	24.1
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	23.5	24.9	24.2
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	22.2	23.6	22.9

Table A.11: Proportion of configurations solved with their VBS, CT and AE statistics for the 1-HUC^{NL} problem for small and large volumes (instances A-S-1, A-S-2, B-S-1 and B-S-2)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
$S \in \{0.2, 0.36\}$	$M_{5PL\text{-}max}$	100.0	103.34	186.38	144.86	0.2	0.4	0.3
	$M_{5PL\text{-}bin}$	100.0	15.28	28.47	21.88	0.2	0.4	0.3
	$M_{2D\text{-}poly}$	100.0	0.43	0.69	0.56	0.8	3.9	2.4
	M_{op}	100.0	1.47	2.25	1.86	0.3	0.4	0.3
	M_{bilin}	100.0	0.03	0.05	0.04	24.3	26.1	25.2
	$M_{PWL\text{-}3}$	100.0	0.4	7.06	3.73	1.3	3.5	2.4
	$M_{PWL\text{-}2}$	100.0	0.09	0.18	0.14	5.8	6.5	6.2
	$M_{PWL\text{-}1}$	100.0	0.01	0.02	0.01	11.8	67.9	39.9
$S \in \{0.002, 0.0036\}$	$M_{5PL\text{-}max}$	100.0	123.58	131.2	127.39	0.5	0.8	0.7
	$M_{5PL\text{-}bin}$	100.0	7.2	16.72	11.96	0.5	0.8	0.7
	$M_{2D\text{-}poly}$	100.0	0.26	0.36	0.31	3.7	17.5	10.6
	M_{op}	100.0	0.18	0.43	0.3	0.5	1.0	0.8
	M_{bilin}	100.0	0.05	0.06	0.06	9.6	29.8	19.7
	$M_{PWL\text{-}3}$	100.0	0.05	0.11	0.08	2.7	171.9	87.3
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	10.8	173.6	92.2
	$M_{PWL\text{-}1}$	100.0	0.01	0.02	0.01	86.3	100.9	93.6

Table A.12: Proportion of configurations solved with their VBS, CT and AE statistics for the fixed-head 1-HUC^{NL} problem for small and large volumes (instances A-S-1, A-S-2, B-S-1 and B-S-2)

Instances	Model	%S	min-CT	max-CT	avg-CT	min-AE	max-AE	avg-AE
$S \in \{0.2, 0.36\}$	$M_{5PL\text{-}max}$	100.0	0.11	0.12	0.11	10.7	20.6	15.7
	$M_{5PL\text{-}bin}$	100.0	0.25	0.4	0.33	10.7	20.6	15.7
	$M_{2D\text{-}poly}$	100.0	0.06	0.08	0.07	14.6	27.3	20.9
	M_{op}	100.0	0.0	0.01	0.01	10.3	21.0	15.7
	M_{bilin}	100.0	0.0	0	0.0	41.0	66.6	53.8
	$M_{PWL\text{-}3}$	100.0	0.02	0.02	0.02	20.2	21.9	21.0
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	19.9	21.8	20.9
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	21.9	22.2	22.0
$S \in \{0.002, 0.0036\}$	$M_{5PL\text{-}max}$	100.0	0.14	0.18	0.16	0.1	1.2	0.7
	$M_{5PL\text{-}bin}$	100.0	0.72	0.84	0.78	0.2	1.0	0.6
	$M_{2D\text{-}poly}$	100.0	0.09	0.09	0.09	4.2	16.7	10.4
	M_{op}	100.0	0.0	0	0.0	0.2	1.1	0.7
	M_{bilin}	100.0	0.0	0	0.0	9.8	30.3	20.1
	$M_{PWL\text{-}3}$	100.0	0.03	0.03	0.03	0.8	176.3	88.6
	$M_{PWL\text{-}2}$	100.0	0.01	0.01	0.01	0.8	175.8	88.3
	$M_{PWL\text{-}1}$	100.0	0.01	0.01	0.01	0.7	177.8	89.2

B Appendix for the polyhedral study of the Symmetric-weight Chain Precedence Knapsack Problem

Table of contents

B.1 Proofs and lemmas

B.1.1 Proof of Property 1

We first recall the downlifting described in [64], before showing that for the SCPKP, it can only yield coefficients $\alpha_{ij} = 0$. To do so, we introduce further definitions. Let U be a MIC. We define π a $pfrs$-order (precedence first, remaining second), an order for items $(i, j) \in U_p \cup U_r$, such that $\pi_{ij} < \pi_{i'j'}$ if one of the following stands:

- $(i, j) \in U_p$ and $(i', j') \in U_r$
- $(i, j) \in U_p$, $(i', j') \in U_p$, $i = i'$ and $j > j'$ (reversed topological order)

- $(i,j) \in U_r$, $(i',j') \in U_r$, $i = i'$ and $j < j'$ (topological order)

We denote $s^\pi(ij)$ (resp. $p^\pi(ij)$) the set of all items (i',j') such that $\pi_{ij} < \pi_{i'j'}$ (resp. $\pi_{ij} > \pi_{i'j'}$).

For a given $(i,j) \in U$, coefficient α_{ij} is computed as follows:

$$\begin{aligned}
\alpha_{ij} = |U| - 1 - \max & \sum_{(i',j') \in U} x_{i'j'} + \sum_{(i',j') \in U_p \cap p^\pi(ij)} \alpha_{i'j'}(1 - x_{i'j'}) \\
\text{s.t.} \quad & \sum_{i'=1}^{I} \sum_{j'=1}^{J} W_{j'} x_{i'j'} \leq C \\
& x_{i'j'} \leq x_{i'j'-1} \qquad \forall x_{i'j'} \in \mathcal{V}, j' \geq 2 \\
& x_{i'j'} = 0 \qquad \forall (i',j') \in U_r, \\
& x_{i'j'} = 1 \qquad \forall (i',j') \in U_p \cap s^\pi(ij), \\
& x_{ij} = 0 \\
& x_{i'j'} \in \{0,1\} \qquad \forall x_{i'j'} \in \mathcal{V}.
\end{aligned}$$

Proof : Consider x_{ij} such that $\pi_{ij} = 1$. Clearly, $p^\pi(ij) = \emptyset$, hence one maximizes

$$\sum_{(i',j') \in U} x_{i'j'}.$$

By definition of a MIC, for each pair of items in U, they must be incomparable, i.e., there cannot be two items of a same group. Also, by definition of the $pfrs$-order, $(i,j+1) \in U$. Hence, as $x_{ij} = 0$, then $x_{ij+1} = 0$, but for all items $(i',j') \in U_p$, then $x_{i'j'} = 1$.

By definition of a MIC, there is a feasible solution with all variables of $U \cup U_p \setminus \{x_{ij+1}\}$ to 1. Consequently, the optimal solution of the problem to compute value α_{ij} has value $|U| - 1$, and coefficient $\alpha_{ij} = 0$.

One can repeat the same reasoning for all variables in U_p. ■

Example 33

Let (4, 4, [5, 4, 2, 2], V , 20) be an instance of the SCPKP. Set $U = \{(1,3),(2,3)\}$ *is a MIC, with* $U_p = \{(1,1), (1,2), (2,1), (2,2)\}$ *and* U_r *contains all other items. Consider the following* $pfrs$*-order:* $\pi = \{(1,2),(1,1),(2,2),(2,1),\ldots\}$. *We do not describe the complete* $pfrs$*-order, as only the order of items in* U_r *is relevant for the purpose of this example. We now proceed to downlift inequality* $x_{13} + x_{23} \leq 1$. *In order to compute* α_{12}*, we solve the following problem:*

$$\begin{aligned}
\alpha_{12} = |U| - 1 - \max \quad & x_{23} \\
s.t. \quad & \sum_{i'=1}^{I} 5x_{i'1} + 4x_{i'2} + 2x_{i'3} + 2x_{1'4} \leq 20 \\
& x_{i'j'} \leq x_{i'j'-1} \qquad \forall x_{i'j'} \in \mathcal{V}, j' \geq 2 \\
& x_{i'j'} = 0 \qquad \forall (i',j') \in U_r,
\end{aligned}$$

$$
\begin{aligned}
& x_{11} = x_{21} = x_{22} = 1, \\
& x_{12} = 0, \\
& x_{i'j'} \in \{0,1\} && \forall x_{i'j'} \in \mathcal{V}.
\end{aligned}
$$

As $x_{12} = 0$, then $x_{13} = 0$ due to chain precedence constraints. The only variable that is not fixed is x_{23}. It is possible to have $x_{23} = 1$, as the total weight is $W_1 + W_1 + W_2 + W_3 = 16 \leq 20$. Consequently $\alpha_{12} = 2 - 1 - 1 = 0$. Same result is obtain for any item is U_p.

B.1.2 Proof of Property 11

Proof : Let (i,j) be an item such that there is $x_{ij'} \in \mathcal{X}$ with $j' > j$. Assume without loss of generality that j' is such that there are no $x_{ij''} \in \mathcal{X}$, $j < j'' < j'$. As $\mathcal{P}$ is a flexible pattern, condition (iii) holds and the property is verified for item (i,j') as $x_{ij'} \in \mathcal{X}$. Let $\mathcal{Y}'$ be the set of variables verifying condition (iii) for item (i,j'). Let $\mathcal{Z}$ be the subset of $\mathcal{Y}'$ with all variables $x_{ij''} \in \mathcal{Y}'$, $j'' \geq j$. One can build $\mathcal{Y} = \mathcal{Y}' \setminus \mathcal{Z} \cup \{x_{ij}\}$. We can prove that the set of variables $\mathcal{Y}$ verifies the property for item (i,j). By construction, $x_{ij} \in \mathcal{Y}$ and $x_{ij''} \notin \mathcal{Y}$ for every $j'' > j$. As there are no $x_{ij''} \in \mathcal{X}$, $j < j'' < j'$ and $\mathcal{Y}'$ is a k-intersection of $\mathcal{X}$ by condition (iii), then $|\mathcal{Y} \cap \mathcal{X}| = k$. Also, as the weights are non-negative, and solution $X_{\mathcal{Y}'}$ is valid by condition (iii), then $X_{\mathcal{Y}}$ is also valid. Consequently, $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$ and the property is verified for x_{ij}.

Let (i,j) be an item such that for all $j' > j$, $x_{ij'} \notin \mathcal{X}$. As $\mathcal{P}$ is a flexible pattern, condition (ii) hold and the property is verified for item (i,J). The proof is the same as in the first case, with $\mathcal{Y}'$ the set of variables verifying condition (ii) for item (i,J).

Thus, for any (i,j), there exist a feasible solution with $x_{ij} = 1$, $x_{ij+1} = 0$ and a total of k variables of $\mathcal{X}$ to 1. ■

B.1.3 Proof of Property 12

Proof Proof of Property 12 : Let $\mathcal{P}$ be a flexible pattern. Suppose that there is $i \leq I$ such that $S_i(\mathcal{Q}_1)$ does not contain the $|S_i(\mathcal{P})| - \underline{U}$ smallest indices of $S_i(\mathcal{P})$. By definition of $\mathcal{Q}_1$, $S_1(\mathcal{Q}_1) = S_1(\mathcal{P})$, meaning the property is trivially verified for $i = 1$. In the following we consider $i > 1$. Let $\mathcal{X} \in \chi(\mathcal{P})$ be a variable set. Let $\mathcal{Y} \in \chi(\mathcal{Q}_1)$ be a variable set, to which we add x_{1J} if $x_{1J} \notin \mathcal{X}$. The solution $X_{\mathcal{Y}}$ is feasible as proven in **Lemma 1**. Let $\mathcal{Y}'$ be the variables set $\mathcal{Y}$ from which we remove all variables of group 1, and to which we add all variables of group i in $\mathcal{X} \setminus \mathcal{Y}$. Because of the symmetric weights, selecting every item $(1,j)$, $j \leq J$ is at least as heavy as selecting items (i,j), $j \leq \max(S_i(\mathcal{P}))$. Consequently:

$$\sum_{x_{i'j'} \in \mathcal{Y}'} s_{i'j'}(\mathcal{Y}') \leq \sum_{x_{i'j'} \in \mathcal{Y}} s_{i'j'}(\mathcal{Y}).$$

As solution $X_{\mathcal{Y}}$ is valid, solution $X_{\mathcal{Y}'}$ must also be valid. By construction, there are $\underline{U}$ variables of group 1 in $\mathcal{Y} \setminus \mathcal{Y}'$. Also, as $|S_i(\mathcal{Q}_1)| < |S_i(\mathcal{P})| - \underline{U}$ by hypothesis, there are at least $\underline{U} + 1$ variables of group i in $\mathcal{Y}' \setminus \mathcal{Y}$.

Besides groups 1 and i, sets $\mathcal{Y}$ and $\mathcal{Y}'$ are identical, hence $|\mathcal{Y}'| > |\mathcal{Y}|$. As $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$, then $\mathcal{Y}'$ is at least a $k+1$-intersection of $\mathcal{X}$, which contradicts the rank k of $\mathcal{P}$.

Thus, $\mathcal{Q}_1$ must contain the $|S_i(\mathcal{P})| - \underline{U}$ smallest indices of $S_i(\mathcal{P})$ for every $i \leq I$. ■

B.1.4 Lemmas for Property 13

Lemma 5

Let $\mathcal{P}$ be a flexible pattern. Let i be an integer belonging to $\{2,\dots,I\}$. For any $u \in \{1,\dots,\underline{U}\}$, if $S_i(\mathcal{P})[u] \in S_i(\mathcal{Q}_1)$ and $|S_i(\mathcal{P})| \geq u + \underline{U}$ then $S_1(\mathcal{P})[u] \geq S_i(\mathcal{P})[u + \underline{U}]$.

The idea of the following proof is illustrated by **Example 34** and **Figure B.1**.

Example 34

*Let $(4, 5, W, V, C)$ be an instance of the SCPKP. Let $\mathcal{P} = \{S_1(\mathcal{P}) = \{1,2\}, S_2(\mathcal{P}) = \{1,3\}, S_3(\mathcal{P}) = \{1,2,3\}, S_4(\mathcal{P}) = \{2,3,4,5\}\}$ be a flexible pattern of rank 7 and with $\underline{U} = 2$. Suppose $\mathcal{Q}_1 = \{\{1,2\},\{1\},\emptyset,\{2,3,4,5\}\}$. In this case $\mathcal{P}$ does not follow **Lemma 5**. Indeed, with $u = 2$, there is $S_1(\mathcal{P})[2] = 1 < S_4(\mathcal{P})[2+2] = 2$.*

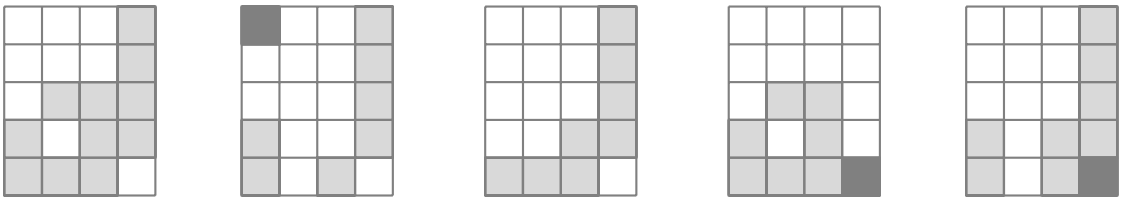
*Let $\mathcal{X} = \{x_{11}, x_{12}, x_{21}, x_{23}, x_{31}, x_{32}, x_{33}, x_{42}, x_{43}, x_{44}, x_{45}\} \in \chi(\mathcal{P})$ be a variable set as represented in **Figure B.1a**. Let $\mathcal{Y} = \{x_{11}, x_{12}, x_{31}, x_{42}, x_{43}, x_{44}, x_{45}\} \in \chi(\mathcal{Q}_1)$ be a variable set. As $\mathcal{P}$ is a flexible pattern, **Lemma 1** proves that solution $X_{\mathcal{Y}\cup\{x_{15}\}}$, as represented in **Figure B.1b**, is feasible. Let $\mathcal{Z} = \{x_{21}, x_{32}\}$ be a variable set. By construction $|\mathcal{Z}| = u = 2$. Clearly, solution $X_{\mathcal{Y}\setminus\{x_{12}\}\cup\mathcal{Z}}$, as represented in **Figure B.1c**, is unfeasible by the rank of $\mathcal{P}$. Hence, $W_2 + W_1 > W_2 + W_3 + W_4 + W_5$.*

*As $\mathcal{P}$ is a flexible pattern, $\mathcal{P}$ verifies **Property 11**. Hence there is a set $\mathcal{Y}'$ such that $x_{41} \in \mathcal{Y}'$, $x_{4j} \notin \mathcal{Y}'$ for every $j \in \{2,\dots,5\}$ and $|\mathcal{Y}' \cap \mathcal{X}| = 7$ and $X_{\mathcal{Y}'}$ is a valid solution. Suppose in this example $\mathcal{Y}' = \{x_{11}, x_{12}, x_{21}, x_{23}, x_{31}, x_{32}, x_{33}, x_{41}\}$ represented in **Figure B.1d**.*

*Note that $S_4(\mathcal{P})[u + \underline{U}] - 1 = S_4(\mathcal{P})[4] - 1 = 1$, and by construction, $x_{4j} \notin \mathcal{Y}'$ for every $j \in \{1,\dots,5\}$. Hence, there are $u + \underline{U} = 4$ variables, namely $x_{21}, x_{23}, x_{32}, x_{33}$, in $\mathcal{Y}' \setminus \mathcal{Y}$. **Property 12** indicates that for every $i \leq I$, $S_i(\mathcal{P})[\underline{U}+1] \in S_i(\mathcal{Q}_1)$. Thus, these 4 variables in $\mathcal{Y}' \setminus \mathcal{Y}$ are split into a set $\mathcal{O}_2 \subset \mathcal{X}$ of 1 to $\underline{U} = 2$ variables in a group, and a set $\mathcal{O}_1 \subset \mathcal{X}$ of $u = 2$ to $u + \underline{U} - 1 = 3$ variables in the other groups. In this case, we arbitrarily chose $\mathcal{O}_2 = \{x_{32}, x_{33}\}$ and $\mathcal{O}_1 = \{x_{21}, x_{23}\}$. Because both $\mathcal{Z}$ and $\mathcal{O}_1$ are subsets of $\mathcal{X} \setminus \mathcal{Y}$ and $|\mathcal{Z}| \leq |\mathcal{O}_1|$, then $W_1 + W_2 + W_3 \geq W_1 + W_2 > W_2 + W_3 + W_4 + W_5$. Hence, with $\mathcal{Y}'' = \mathcal{Y}' \setminus \mathcal{O}_1 \cup \{x_{42}, x_{43}, x_{44}, x_{45}\}$ represented in **Figure B.1e**, as solution $X_{\mathcal{X}'}$ is feasible, then solution $X_{\mathcal{Y}''}$ is also feasible. However $\mathcal{Y}''$ is an 8-intersection of $\mathcal{X}$, which contradicts the rank of $\mathcal{P}$.*

Proof : Let $\mathcal{P}$ be a flexible pattern of rank k and $\mathcal{Q}_1$ a lower sub-pattern of $\mathcal{P}$. Suppose that there is $u \leq \underline{U}$ and $i > 1$ such that $S_1(\mathcal{P})[u] \in S_i(\mathcal{Q}_1)$ and $|S_i(\mathcal{P})| \geq u + \underline{U}$ but $S_1(\mathcal{P})[u] < S_i(\mathcal{P})[u + \underline{U}]$.

Let $\mathcal{X} \in \chi(\mathcal{P})$ and $\mathcal{Y} \in \chi(\mathcal{Q}_1)$ be variable sets. By **Lemma 1**, solution $X_{\mathcal{Y}\cup\{x_{1J}\}}$ is feasible. However, for a variable set $\mathcal{Z}$, $\mathcal{Z} \subseteq \mathcal{X} \setminus \mathcal{Y}$, it is not possible to create a feasible solution from $X_{\mathcal{Y}}$ by setting every variable of $\mathcal{Z}$ to 1, and $|\mathcal{Z}| - 1$ variables of $\mathcal{Y}$ to 0. Otherwise there would be $k+1$ variables of $\mathcal{X}$ to 1, which contradicts



(a) Set $\mathcal{X}$ (b) Set $\mathcal{Y} \cup x_{45}$ (c) Non feasible set (d) Set $\mathcal{Y}'$ (e) Set $\mathcal{Y}''$

Figure B.1: Illustration of **Example 34**

the rank of $\mathcal{P}$. With $|\mathcal{Z}| = u$, as $X_{\mathcal{Y}\cup\{x_{1J}\}}$ is a feasible solution, we deduce the following:

$$\sum_{x_{i'j'}\in\mathcal{Z}} s_{i'j'}(\mathcal{X}) > \sum_{j=S_1(\mathcal{P})[u]+1}^{J} W_j.$$

Because $\mathcal{P}$ is flexible, **Property 11** implies that for $j = S_i(\mathcal{P})[u + \underline{U}]$ there exists a set $\mathcal{Y}' \in \mathcal{V}$ being a k-intersection of $\mathcal{X}$ with $x_{ij-1} \in \mathcal{Y}'$ and $x_{ij'} \notin \mathcal{Y}'$, $j' \geq j$, such that $X_{\mathcal{Y}'}$ is valid. From **Property 4** as $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$, there is $\mathcal{Y}' \subseteq \mathcal{Y}$ a k-intersection of $\mathcal{X}$. Hence we suppose without loss of generality that $|\mathcal{Y}| = k$.

In the case $S_i(\mathcal{Q}_1) = S_i(\mathcal{P})$, clearly there are $u + \underline{U}$ more variables of group i in $\mathcal{Y} \setminus \mathcal{Y}'$. As $card(\mathcal{Q}_1) = k$ and $\mathcal{Y} \in \mathcal{X}(\mathcal{Q}_1)$, then $|\mathcal{Y}| = |\mathcal{Y}'| = k$. Hence, there are $u + \underline{U}$ variables of groups different than i in $\mathcal{Y}' \setminus \mathcal{Y}$. **Property 12** indicates that for each $i' \leq I$, $|S_{i'}(\mathcal{Q}_1)| \geq |S_{i'}(\mathcal{P})| - \underline{U}$. Thus, these $u + \underline{U}$ variables in $\mathcal{Y}' \setminus \mathcal{Y}$ are split into a set $\mathcal{O}_2 \subset \mathcal{X}$ of 1 to $\underline{U}$ variables in a group, and a set $\mathcal{O}_1 \subset \mathcal{X}$ of u to $u + \underline{U} - 1$ variables in the other groups.

In the case $S_i(\mathcal{Q}_1) \subset S_i(\mathcal{P})$ with $S_i(\mathcal{P})[u] \in S_i(\mathcal{Q}_1)$ there are between $\underline{U} + 1$ and $u + \underline{U} - 1$ variables of group i in $\mathcal{Y} \setminus \mathcal{Y}'$ and reciprocally between $\underline{U} + 1$ and $u + \underline{U} - 1$ variables of groups different than i in $\mathcal{Y}' \setminus \mathcal{Y}$. In this case we consider $\mathcal{O}_1 \subset \mathcal{X}$ to be the set of $\underline{U} + 1$ to $u + \underline{U} - 1$ variables if $\mathcal{Y}' \setminus \mathcal{Y}$, and $\mathcal{O}_2 = \emptyset$.

We know that $\mathcal{Z} \in \mathcal{X}$ and $\mathcal{Z} \cap \mathcal{Y} = \emptyset$ and similarly, $\mathcal{O}_1 \subset \mathcal{X}$ and $\mathcal{O}_1 \cup \mathcal{Y} = \emptyset$. Also, we know that $|\mathcal{O}_1| \geq u = |\mathcal{Z}|$. We deduce that:

$$\sum_{x_{i'j'}\in\mathcal{O}_1} s_{i'j'}(\mathcal{X}) \geq \sum_{x_{i'j'}\in\mathcal{Z}} s_{i'j'}(\mathcal{X}) > \sum_{j=S_1(\mathcal{P})[u]+1}^{J} W_j.$$

By construction, $S_i(\mathcal{P})[1] \leq J$ and by hypothesis, $S_i(\mathcal{P})[u + \underline{U}] > S_1(\mathcal{P})[u]$, which means that:

$$\sum_{j\in S_i(\mathcal{P}):j\geq S_i(\mathcal{P})[u+\underline{U}]} s_{ij}(\mathcal{X}) = \sum_{j=S_j(\mathcal{P})[u+\underline{U}]}^{S_i(\mathcal{P})[1]} W_j \leq \sum_{j=S_1(\mathcal{P})[u]+1}^{J} W_j.$$

The $u + \underline{U}$ variables of group i that are in $\mathcal{Y}$ but not in $\mathcal{Y}'$ are lighter than the ones in $\mathcal{Z}$, thus lighter than variables of $\mathcal{O}_1$. Hence we build $\mathcal{Y}'' = \mathcal{Y}' \setminus \mathcal{O}_1$ to which we add all the $u + \underline{U}$ variables of group i in $\mathcal{X} \setminus \mathcal{Y}'$. As $X_{\mathcal{Y}'}$ is valid, then $X_{\mathcal{Y}''}$ is also valid. However $|\mathcal{O}_1| \leq u + \underline{U} - 1$, meaning that the new solution has at least $k + 1$ variables of $\mathcal{X}$ to 1, which contradicts the rank of $\mathcal{P}$. ■

Lemma 6

Let $\mathcal{P}$ be a flexible pattern and $\mathcal{Q}_1$ a lower sub-pattern of $\mathcal{P}$. Let i be an integer belonging to $\{2, \ldots, I\}$.

For any $u \in \{1,\dots,\underline{U}\}$, if $S_i(\mathcal{P})[u] \notin S_i(\mathcal{Q}_1)$ and $|S_i(\mathcal{P})| \geq u + \underline{U}$ then $S_1(\mathcal{P})[u] \geq S_i(\mathcal{P})[u + \underline{U}]$.

Proof : Suppose that there is $u \leq \underline{U}$ and $i > 1$ such that $S_i(\mathcal{P})[u] \notin S_i(\mathcal{Q}_1)$ and $|S_i(\mathcal{P})| \geq u + \underline{U}$ but $S_1(\mathcal{P})[u] < S_i(\mathcal{P})[u + \underline{U}]$.

Let $\mathcal{X} \in \chi(\mathcal{P})$ and $\mathcal{Y} \in \chi(\mathcal{Q}_1)$ be variable sets. Because $\mathcal{P}$ is flexible, **Lemma 1** proves that solution $X_{\mathcal{Y}\cup\{x_{1J}\}}$ is valid. By definition, $S_i(\mathcal{P})[1] \leq J$ and $S_i(\mathcal{P})[\underline{U}+1] \geq S_i(\mathcal{P})[u+\underline{U}]$ and by hypothesis $S_i(\mathcal{P})[u+\underline{U}] > S_1(\mathcal{P})[u]$, which means that:

$$\sum_{j=S_i(\mathcal{P})[\underline{U}+1]+1}^{S_i(\mathcal{P})[1]} W_j \leq \sum_{j=S_1(\mathcal{P})[u]+1}^{J} W_j.$$

Hence, one can build $\mathcal{Y}' = \mathcal{Y} \cup \{x_{1J}\}$ from which we remove all variables $x_{1j} \in \mathcal{X}$, $j > S_1(\mathcal{P})[u]$ and to which we add all variables for group i: $x_{ij} \in \mathcal{X} \setminus \mathcal{Y}$, with solution $X_{\mathcal{Y}'}$ being valid. However, there are $u-1$ variables $x_{1j} \in \mathcal{X}$ such that $j > S_1(\mathcal{P})[u]$, and at least u variables $x_{ij} \in \mathcal{X} \setminus \mathcal{Y}$ as $S_i(\mathcal{P})[u] \notin S_i(\mathcal{Q}_1)$. As $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$, then $\mathcal{Y}'$ is at least a $k+1$-intersection of $\mathcal{X}$ which contradicts the rank of $\mathcal{P}$. ■

B.1.5 Lemmas for Property 14

Lemma 7

Let $\mathcal{P}$ be a flexible pattern of rank k and $\mathcal{Q}_1$ a lower sub-pattern of $\mathcal{P}$. Let $\mathcal{Q}_i$ be a lower sub-pattern of $\mathcal{P}$ with $i > 1$. If $|S_1(\mathcal{Q}_1)| + |S_i(\mathcal{Q}_1)| = |S_1(\mathcal{Q}_i)| + |S_i(\mathcal{Q}_i)|$ and $|S_i(\mathcal{P})| \geq 2\underline{U}$, then for each $i \leq I$, there is $\mathcal{Q}_i$ such that for every $i' \leq I$ with $|S_{i'}(\mathcal{P})| \geq 2\underline{U}$, $S_{i'}(\mathcal{P})[2\underline{U}] \in S_{i'}(\mathcal{Q}_i)$.

Proof : We are in the case $|S_1(\mathcal{Q}_1)| + |S_i(\mathcal{Q}_1)| = |S_1(\mathcal{Q}_i)| + |S_i(\mathcal{Q}_i)|$. Let $\mathcal{L} = \{1,..,I\} \setminus \{1,i\}$ be a set of indices. By definition, $card(\mathcal{Q}_1) = card(\mathcal{Q}_i) = k$ hence we deduce the following equation:

$$\sum_{i' \in \mathcal{L}} |S_{i'}(\mathcal{Q}_1)| = \sum_{i' \in \mathcal{L}} |S_{i'}(\mathcal{Q}_i)|.$$

We suppose without loss of generality that $S_{i'}(\mathcal{Q}_1) = S_{i'}(Q_i)$ for each $i' \in \mathcal{L}$. Indeed, by definition $\mathcal{Q}_1$ and $\mathcal{Q}_i$ minimize the weight of their respective variable sets. Hence they have both the exact same weight for their variables sets restricted to indices in $\mathcal{L}$, otherwise it is clear that one of them do not minimize the weight of its variable sets. Consequently, one can modify $\mathcal{Q}_i$ such that $S_{i'}(\mathcal{Q}_1) = S_{i'}(Q_i)$ for each $i' \in \mathcal{L}$ without increasing the weight of the variable sets of $\mathcal{Q}_i$.

Property 12 indicates that $S_{i'}(\mathcal{P})[\underline{U}+1] \in S_{i'}(\mathcal{Q}_1)$. As $\mathcal{P}$ is flexible pattern, $\underline{U} \geq 1$ by condition (i), and $\underline{U}+1 \geq 2\underline{U}$. Clearly the Lemma is verified for $\mathcal{Q}_1$.

As $S_{i'}(\mathcal{Q}_1) = S_{i'}(\mathcal{Q}_i)$ for any $i' \in \mathcal{L}$, the Lemma is also verified for any $S_{i'}(\mathcal{Q}_i)$ with $i' \in \mathcal{L}$. By definition of $\mathcal{Q}_i$, $S_i(\mathcal{Q}_i) = S_i(\mathcal{P})$, trivially verifying the Lemma. As $|S_1(\mathcal{P})| = \underline{U} < 2\underline{U}$, the Lemma does not concern $S_1(\mathcal{Q}_i)$. Thus, the Lemma is verified for any set $S_{i'}(\mathcal{Q}_i)$ for which $|S_{i'}(\mathcal{Q}_i)| \geq 2\underline{U}$. ■

Lemma 8

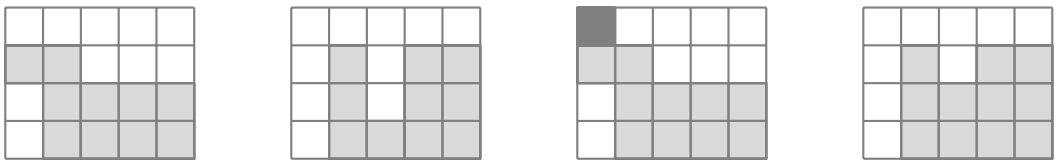
Let $\mathcal{P}$ be a flexible pattern of rank k. Let $\mathcal{Q}_i$ be a lower sub-pattern of $\mathcal{P}$ with $i > 1$. If $|S_1(\mathcal{Q}_i)| = 0$, then for each $i' \leq I$ with $|S_{i'}(\mathcal{P})| \geq 2\underline{U}$, $S_{i'}(\mathcal{P})[2\underline{U}] \in S_{i'}(\mathcal{Q}_i)$.

The idea of the following proof is illustrated by **Example 35** and **Figure B.2**.

Example 35

*Let $(5, 3, W, V, C)$ be an instance of the SCPKP. Let $\mathcal{P} = \{S_1 = \{3\}, S_2 = \{1,2,3\}, S_3 = \{1,2,3\}, S_4 = \{1,2,3\}, S_5 = \{1,2,3\}\}$ be a flexible pattern of rank 10. The smallest set of $\mathcal{P}$ is $S_1(\mathcal{P})$ hence $\underline{U} = 1$. Suppose $\mathcal{Q}_1 = \{S_1 = \{3\}, S_2 = \{1,2,3\}, S_3 = \{1,2\}, S_4 = \{1,2\}, S_5 = \{1,2\}\}$ and $\mathcal{Q}_2 = \{S_1 = \emptyset, S_2 = \{1,2,3\}, S_3 = \{1\}, S_4 = \{1,2,3\}, S_5 = \{1,2,3\}\}$. We are in the case of **Lemma 8** as $|S_1(\mathcal{Q}_2)| = 0$, but not in the case of **Lemma 7**, as $|S_1(\mathcal{Q}_1)| + |S_2(\mathcal{Q}_1)| = 4 \neq |S_1(\mathcal{Q}_2)| + |S_2(\mathcal{Q}_2)| = 3$. However, **Lemma 8** is not verified, as $S_3(\mathcal{P})[2\underline{U}] = S_3(\mathcal{P})[2] = 2 \notin S_3(\mathcal{Q}_2)$.*

*Let $\mathcal{Y} = \{x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{41}, x_{42}, x_{51}, x_{52}\} \in \mathcal{X}(\mathcal{Q}_1)$ and $\mathcal{Y}' = \{x_{21}, x_{22}, x_{23}, x_{31}, x_{41}, x_{42}, x_{43}, x_{51}, x_{52}, x_{53}\} \in \mathcal{X}(\mathcal{Q}_2)$ be variable sets. As $\mathcal{P}$ is flexible-pattern, **Lemma 1** proves that solution $X_{\mathcal{Y}\cup\{x_{14}\}}$ is valid. Also, $\mathcal{Q}_2$ minimizes the sum of the weights, $\{x_{43}, x_{53}\} \in \mathcal{Y}'$ and $\{x_{32}, x_{33}\} \notin \mathcal{Y}'$ hence $W_3 + W_3 \leq W_2 + W_3$. Clearly, $W_3 + W_3 \leq W_2 + W_3 \leq W_1 + W_2 + W_3 + W_4$. With $\mathcal{O} = \mathcal{Y}' \setminus \mathcal{Y} = \{x_{43}, x_{53}\}$, the set $\mathcal{Y}'' = \mathcal{Y} \cup \{x_{14}\} \setminus \{x_{13}, x_{14}\} \cup \mathcal{O}$ yields a feasible solution $X_{\mathcal{Y}''}$. However, $\mathcal{Y}''$ is a 9-intersection of $\mathcal{X}$, which contradicts the rank of $\mathcal{P}$.*



(a) Set $\mathcal{Y}$ (b) Set $\mathcal{Y}'$ (c) Set $\mathcal{Y} \cup \{x_{14}\}$ (d) Set $\mathcal{Y}''$

Figure B.2: Illustration of **Example 35**

Proof : Because of the shape of $\mathcal{Q}_1$ (see **Property 12**), $|S_1(\mathcal{Q}_1)| = \underline{U}$ meaning that $|S_i(\mathcal{Q}_1)| \geq |S_i(\mathcal{P})| - \underline{U}$. We first consider the case $|S_i(\mathcal{Q}_1)| = |S_i(\mathcal{P})| - \underline{U}$. In such case, $|S_i(\mathcal{Q}_1)| + |S_1(\mathcal{Q}_1)| = |S_i(\mathcal{P})|$. By definition of $\mathcal{Q}_i$, $|S_i(\mathcal{Q}_i)| = S_i(\mathcal{P})$. As we are in the case $|S_1(\mathcal{Q}_i)| = 0$, then $|S_i(\mathcal{Q}_i)| + |S_1(\mathcal{Q}_i)| = |S_i(\mathcal{Q}_1)| + |S_1(\mathcal{Q}_1)|$ which is covered in **Lemma 7**.

Hence, we consider $|S_i(\mathcal{Q}_1)| \geq |S_i(\mathcal{P})| - \underline{U} + 1$. Consequently, $|S_i(\mathcal{Q}_1)| + |S_1(\mathcal{Q}_1)| \geq |S_i(\mathcal{Q}_i)| + |S_1(\mathcal{Q}_i)| + 1$. Suppose there is $i' \neq 1$, $i' \neq i$ such that $S_{i'}(\mathcal{P})[2\underline{U}] \notin S_{i'}(\mathcal{Q}_i)$. With $v \in [0, |S_{i'}(\mathcal{P})| - 2\underline{U}]$, we have $|S_{i'}(\mathcal{Q}_i)| = |S_{i'}(\mathcal{P})| - 2\underline{U} - v$, i.e., $S_{i'}(\mathcal{P})[2\underline{U} + v] \notin S_{i'}(\mathcal{Q}_i)$. We know by the shape of $\mathcal{Q}_1$ that $|S_{i'}(\mathcal{Q}_1)| \geq |S_{i'}(\mathcal{P})| - \underline{U}$ and thus $|S_{i'}(\mathcal{Q}_1)| \geq |S_{i'}(\mathcal{Q}_i)| + \underline{U} + v$. This leads to the following result:

$$|S_1(\mathcal{Q}_1)| + |S_i(\mathcal{Q}_1)| + |S_{i'}(\mathcal{Q}_1)| - |S_1(\mathcal{Q}_i)| - |S_i(\mathcal{Q}_i)| - |S_{i'}(\mathcal{Q}_i)| \geq \underline{U} + v + 1.$$

There must be a set of indices $\mathcal{L}$, $1 \notin \mathcal{L}$, $i \notin \mathcal{L}$, $i' \notin \mathcal{L}$ such that:

$$\underline{U} + v + 1 \leq \sum_{l \in \mathcal{L}} |S_l(\mathcal{Q}_i)| - |S_l(\mathcal{Q}_1)| \leq 2\underline{U} + v.$$

If not, then there is a set l such that $|S_l(\mathcal{Q}_i)|-|S_l(\mathcal{Q}_1)| > \underline{U}+1$, which contradicts the shape of $\mathcal{Q}_1$ (see **Property 12**) which is that $|S_l(\mathcal{Q}_1)| \geq |S_l(\mathcal{P})| - \underline{U}$.

Let $\mathcal{X} \in \chi(\mathcal{P})$, $\mathcal{Y} \in \chi(\mathcal{Q}_1)$ and $\mathcal{Y}' \in \chi(\mathcal{Q}_i)$ be variable sets. Let $\mathcal{O}$ be the set of variables $x_{lj} \in \mathcal{Y}' \setminus \mathcal{Y}$ with $l \in \mathcal{L}$. Note that $\underline{U}+v+1 \leq |\mathcal{O}| \leq 2\underline{U}+v$. Also, $S_{i'}(\mathcal{P})[2\underline{U}+v] \notin S_{i'}(\mathcal{Q}_i)$, $\mathcal{O}$ is sub-pattern of $\mathcal{Q}_i$ and by definition $\mathcal{Q}_i$ minimizes the sum of the weights and $\mathcal{O}$ is a sub-pattern of $\mathcal{Q}_i$, hence we deduce:

$$\sum_{x_{lj} \in \mathcal{O}} s_{lj} \leq \sum_{j=S_{i'}(\mathcal{P})[2\underline{U}+v+1]+1}^{S_{i'}(\mathcal{P})[1]} W_j \leq \sum_{j=S_{i'}(\mathcal{P})[2\underline{U}+v+1]+1}^{J} W_j \leq \sum_{j=1}^{J} W_j.$$

As $\mathcal{P}$ is flexible pattern, **Lemma 1** provides a feasible solution $X_{\mathcal{Y}\cup\{x_{1J}\}}$. One could create a new feasible solution from $X_{\mathcal{Y}\cup\{x_{1J}\}}$ by setting all variables of group 1 to 0, and all variables of $\mathcal{O}$ to one, creating a lighter solution, thus valid. However, $|\mathcal{O}| \geq \underline{U}+v+1 \geq \underline{U}+1 > |S_1(\mathcal{Q}_1)| = \underline{U}$. The new solution has at least $k+1$ variables of $\mathcal{X}$ to 1, which contradicts the rank of $\mathcal{P}$. ■

Lemma 9

Let $\mathcal{P}$ be a flexible pattern of rank k. Let $\mathcal{Q}_i$ be a lower sub-pattern of $\mathcal{P}$ with $i > 1$. If $|S_1(\mathcal{Q}_i)| > 0$, then for each $i' \leq I$ with $|S_{i'}(\mathcal{P})| \geq 2\underline{U}$, $S_{i'}(\mathcal{P})[2\underline{U}] \in S_{i'}(\mathcal{Q}_i)$.

Proof : We consider the case where $|S_1(\mathcal{Q}_1)|+|S_i(\mathcal{Q}_1)| \neq |S_1(\mathcal{Q}_i)|+|S_i(\mathcal{Q}_i)|$, as otherwise it is the case of **Lemma 7**. Suppose there is $i' \neq 1$, $i' \neq i$ such that $S_{i'}(\mathcal{P})[2\underline{U}] \notin S_{i'}(\mathcal{Q}_i)$. Let $v \in [0, |S_{i'}(\mathcal{P})| - 2\underline{U}]$ be such that $|S_{i'}(\mathcal{Q}_i)| = |S_{i'}(\mathcal{P})| - 2\underline{U} - v$. Let u be the smallest value such that $S_1(\mathcal{P})[u] \in S_1(\mathcal{Q}_i)$, i.e., $u = \underline{U} - |S_1(\mathcal{Q}_i)| + 1$. Let $\mathcal{X} \in \chi(\mathcal{P})$ and $\mathcal{Y} \in \chi(\mathcal{Q}_i)$ be variable sets. We know that $X_{\mathcal{Y}}$ is a feasible solution. We also know by **Property 13** that a flexible pattern $\mathcal{P}$ is such that $S_1(\mathcal{P})[u] \geq S_{i'}(\mathcal{P})[u+\underline{U}]$, which means that:

$$\sum_{j' \leq S_{i'}(\mathcal{P})[u+\underline{U}]} s_{i'j'}(\mathcal{X}) \leq \sum_{j \leq S_1(\mathcal{P})[u]} s_{1j}(\mathcal{X}).$$

Hence, one can build $\mathcal{Y}' = \mathcal{Y}$ from which we remove all variables $x_{1j} \in \mathcal{Y}$ and to which we add all $x_{i'j'} \in \mathcal{X}$, $j' \leq S_{i'}(\mathcal{P})[u+\underline{U}]$. Clearly, as $X_{\mathcal{Y}}$ is a valid solution, so is $X_{\mathcal{Y}'}$. In group i', by hypothesis there are $|S_{i'}(\mathcal{P})| - 2\underline{U} - v$ variables in $\mathcal{Y}$, and by construction $|S_{i'}(\mathcal{P})| - (\underline{U}+u) + 1$ in $\mathcal{Y}'$ Hence in group i', there are $\underline{U}+v-u+1$ variables in $\mathcal{Y}' \setminus \mathcal{Y}$. In group 1, by hypothesis there are $\underline{U}-u+1$ variables in $\mathcal{Y}$, and by construction 0 variables in $\mathcal{Y}'$. For groups different than i' and 1 variable sets $\mathcal{Y}$ and $\mathcal{Y}'$ are identical by construction. We deduce $|\mathcal{Y}'| - |\mathcal{Y}| = \underline{U} + v - u + 1 - (\underline{U} - u + 1) = v \geq 0$.

In the case $v \geq 1$, as $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$, then $\mathcal{Y}'$ is at least a $k+1$-intersection of $\mathcal{X}$, which contradicts the rank of $\mathcal{P}$.

In the case $v = 0$, by construction $\mathcal{Y}$ and $\mathcal{Y}'$ contain the same variables for group i. Also by construction, the weight of $\mathcal{Y}'$ is lighter than or equals to the weight of $\mathcal{Y}$. Clearly, there is a $\mathcal{Q}'_i$ such that $\mathcal{Y}' \in \chi(\mathcal{Q}'_i)$. However, by construction $S_1(\mathcal{Q}_{i'})$ is empty, which cannot be possible as proven by **Lemma 8**. ■

B.1.6 Proof of Theorem 9

The following Lemma aims to use condition (iv) to provide feasible solutions for any variable in $\mathcal{X} \in \chi(\mathcal{P})$. The idea is for these solutions to have at least all variables of $\mathcal{Y}_{U'}$ to 1, with $\mathcal{Y}_{U'} \in \chi(\mathcal{R}_{U'})$. As these solutions have many variables to 1 in common, this will be convenient to prove them to be linearly independent in **Theorem 9**.

Lemma 10

If the nested sub-patterns $\{\mathcal{R}_u, 1 \leq u \leq U'\}$ verify (iv), then for any $x_{ij} \in \mathcal{X} \setminus \mathcal{Y}_1$, there is a $u \leq U'$ such that there is a set $\mathcal{Z}$ containing u variables $x_{ij'} \in \mathcal{X} \setminus \mathcal{Y}_u, j' \leq j$ and $\mathcal{Z} \cup \mathcal{Y}_u$ is a k-intersection of $\mathcal{X}$.

Proof : Let $\mathcal{P}$ be a pattern verifying (iv). Let $\mathcal{X} \in \chi(\mathcal{P})$ and $\mathcal{Y}_u \in \chi(\mathcal{R}_u)$ for each $u \leq U'$ be variable sets.

Firstly, by definition of $\mathcal{R}_1$, there is a feasible solution with all variables of $\mathcal{Y}_1$ to 1, and $x_{ij} = 1$, with $x_{ij} \in \mathcal{X} \setminus \mathcal{Y}_u$, $x_{ij'} \notin \mathcal{X} \setminus \mathcal{Y}_u$, $j' < j$. Hence the Lemma is verified for every $x_{ij} \in \mathcal{X} \setminus \mathcal{Y}_1$ such that $x_{ij'} \notin \mathcal{X} \setminus \mathcal{Y}_1$, $j' < j$.

Secondly, we can define a recursive rule. Let x_{ij} be a variable such that $x_{ij} \in \mathcal{X} \setminus \mathcal{Y}_1$. Suppose for $x_{ij'}$ the Lemma is verified for $\mathcal{R}_u$, with j' such that $x_{ij''} \notin \mathcal{X}$, $j' < j'' < j$. In other word, there is a feasible solution with $x_{ij'}$ to 1 and all variables of $\mathcal{Y}_u$ to 1. Note that $x_{ij} = 0$ in this solution. By definition of the nested sub-pattern $\mathcal{R}_{u+1}$, there is a feasible solution with all variables of $\mathcal{Y}_{u+1}$ and the $u+1$ variables $x_{ij'} \in \mathcal{X} \setminus \mathcal{Y}_u$, $j' \leq j$. We distinguish two cases:

The first case is $|S_i(\mathcal{R}_{u+1})| = |S_i(\mathcal{R}_u)|$. By hypothesis, there is a $\mathcal{Z}$ containing u variables of group i from $\mathcal{X} \setminus \mathcal{Y}_u$, with $x_{ij'} \in \mathcal{Z}$, $x_{ij} \notin \mathcal{Z}$ such that $\mathcal{Z} \cup \mathcal{Y}_u$ is a k-intersection of $\mathcal{X}$. As (iv) holds, then clearly there is $\mathcal{Z}'$ containing $u+1$ variables of group i from $\mathcal{X} \setminus \mathcal{Y}_{u+1}$, such that $\mathcal{Z}' \cup \mathcal{Y}_{u+1}$ is a k-intersection of $\mathcal{X}$. As $x_{ij''} \notin \mathcal{X}$, $j' < j'' < j$, then $x_{ij} \in \mathcal{Z}'$ and $x_{ij''} \notin \mathcal{Z}$ with $j'' \geq j$. Hence the Lemma is verified for x_{ij}.

The second case is $|S_i(\mathcal{R}_{u+1})| = |S_i(\mathcal{R}_u)| - 1$. In which case, with $\mathcal{Z}'$ containing $u+1$ variables of group i from $\mathcal{X} \setminus \mathcal{Y}_{u+1}$, $\mathcal{Z} \cup \mathcal{Y}_u = \mathcal{Z}' \cup \mathcal{Y}_{u+1}$. Consequently $x_{ij} \notin \mathcal{Z}$.

From these two cases, we deduce that if the Lemma is verified for x_{ij}, it is verified for any $x_{ij'} \in \mathcal{X} \setminus \mathcal{Y}_1$, with $j' \leq j$.

Finally, by definition of $\mathcal{R}_{U'}$, there is at most U' variables in $S_i(\mathcal{P}) \setminus S_i(\mathcal{R}_{U'})$. Hence this Lemma is necessarily verified for every $x_{ij} \in \mathcal{X}$ such that $x_{ij'} \notin \mathcal{X}$, $j' > j$.

Because of the initialization with $\mathcal{Y}_1$, the recursive rule between $\mathcal{Y}_u$ and $\mathcal{Y}_{u+1}$, and because for every $x_{ij} \in \mathcal{X}$ such that $x_{ij'} \notin \mathcal{X}$, $j' > j$ the Lemma is verified, then the Lemma is verified for all $x_{ij} \in \mathcal{X} \setminus \mathcal{R}_1$. ■

Proof : Proof of **Theorem 9** Let $\mathcal{X} \in \chi(\mathcal{P})$ and $\mathcal{Y}_u \in \chi(\mathcal{R}_u)$ be variable sets.

The points will be enumerated iteratively.

First, from **Lemma 10**, we know that for every $x_{ij} \in \mathcal{X} \setminus \mathcal{Y}_1$ such that $x_{ij'} \notin \mathcal{X} \setminus \mathcal{Y}_1$, $j' < j$, there is a feasible solution with $x_{ij} = 1$ and with $x_{ij+1} = 0$, with all variables of $\mathcal{Y}_1$ to 1. These solutions are all linearly independent, as for each x_{ij} considered, it is the only solution with $x_{ij} = 1$.

As condition (v) holds, there is a feasible solution with all variables of $\mathcal{Y}_2$ to 1, and two variables $x_{ij}, x_{i'j'} \notin \mathcal{Y}_1$, $i \neq i'$ to 1. This solution is linearly independent to the previously mentioned, as it is the only one with a variable of $\mathcal{Y}_1$ to 0.

From **Lemma 10**, we know that there is a feasible solution with $\mathcal{Y}_2$ and two variables $x_{ij}, x_{ij'} \in \mathcal{X} \setminus \mathcal{Y}_2$, $j' \neq j$. For each solution where both x_{ij} and $x_{ij'}$ are not in $\mathcal{Y}_1$ is linearly independent to the others, as it is the only one with $x_{ij'} = 1$.

One can keep enumerating points with the same process. With **Lemma 10**, $|\mathcal{X}|-|\mathcal{Y}_1|$ linearly independent points are generated. With condition (v), one linearly independent point is generated for every $u \in \{2,\dots,U'\}$, which is a total of $U'-1$ linearly independent points.

With condition (vi), for each $x_{ij} \in \mathcal{Y}_{U'}$, there is a set $\mathcal{Z}$ containing all variables $x_{ij'} \in \mathcal{Y}_{U'}$, $j' < j$ and all variables $x_{i'j'} \in \mathcal{Y}_{U'}$, $i' \neq i$, without x_{ij} and with $\mathcal{Z}$ being a k-intersection of $\mathcal{X}$. Hence, starting with greater j, each new solution is the first one with $x_{ij} = 0$, being linearly independent to the others. With condition (vi), one new feasible solution is created for every $x_{ij} \in \mathcal{Y}_{U'}$, meaning $|\mathcal{Y}_{U'}|$ new linearly independent points.

A total of $|\mathcal{X}| - |\mathcal{Y}_1| + U' - 1 + |\mathcal{Y}_{U'}| = card(\mathcal{P}) - (k-1) + U' - 1 + (k - U')$ linearly independent points are generated, i.e., $card(\mathcal{P})$ linearly independent points. Moreover, from **Theorem 7** for each $x_{ij} \notin \mathcal{X}$, there is a feasible solution with $x_{ij} = 1$ and $x_{ij+1} = 0$ that verifies $(pi(\mathcal{X}))$ to equality. Sequentially adding these points associated with their corresponding solutions to our pool of $card(\mathcal{P})$ points still keeps them linearly independent, as there are the only ones with $x_{ij} = 1$ and $x_{ij+1} = 0$ with $x_{ij} \notin \mathcal{X}$. As there are $n - card(\mathcal{P})$ of these new points, there is a total of n linearly independent points. ■

B.1.7 Proof of Property 19

Proof : First, we consider $\mathcal{P}$ with all sets $S_i(\mathcal{P})$ of cardinality $\underline{U}$. We recall that $|V_{SCPKP}| = |\mathcal{X}| + 2$ by construction of G. For each vertex $x_{(i)(1)}$, there is an arc $(p, x_{(i)(1)})$. For each vertex $x_{(i)(j)}$ with $j > 1$, there is an arc $(x_{(i)(j-1)}, x_{(i)(j)})$. This sums up to a total of $|\mathcal{X}|$ arcs. For each vertex $x_{(i)(j)}$, there is an arc $(x_{(i)(j)}, q)$. Accounting for the previously enumerated arcs, there are $2|\mathcal{X}|$ arcs.

For each $x_{(i)(j)}$, there is an arc $(x_{(i)(j)}, x_{(i')(1)})$, for every $i' > 1$. Consequently, for every $i \leq I$, there are $|S_i(\mathcal{P})| \cdot (I - i) = \underline{U} \cdot (I - i)$ arcs. This yields a total of $\sum_{i=1}^{I} \underline{U} \cdot (I - i) = \sum_{i=0}^{I-1} \underline{U} \cdot i$ which is equal to $\underline{U} \cdot \frac{(I-1)\cdot I}{2}$.

In total, the number of arcs $|A|$ is equal to $2 \cdot \underline{U} \cdot I + \frac{1}{2} \cdot \underline{U} \cdot (I^2 - I)$ which can be approximated by $\underline{U} \cdot I^2$ when $\underline{U}$ and I are large. Also, we can compute $|V_{SCPKP}|^2 = (I \cdot \underline{U} + 2)^2 = I^2 \cdot \underline{U}^2 + 4 \cdot \underline{U} \cdot I + 4 > I^2 \cdot J^2 + 4 \cdot \underline{U} \cdot I$, which can be approximated by $\underline{U}^2 \cdot I^2$ when $\underline{U}$ and I are large, meaning that $\frac{|V_{SCPKP}|^2}{|A|} \approx \underline{U}$.

Consider now that we add one vertex $x_{(i)(j)}$ to G. This adds arc $(x_{(i)(j-1)}, x_{(i)(j)})$, $(x_{(i)(j)}, q)$ and at most $I-1$ arcs $(x_{(i)(j)}, x_{(i'),(1)})$, with $i' > i$. This results to a total of $I+1$ arcs, which is smaller than $|V_{SCPKP}|$, hence by adding one vertex, the factor $\underline{U}$ between $|V_{SCPKP}|^2$ and $|A|$ holds. The same result applies by adding more than one vertex. Consequently, for any pattern $\mathcal{P}$, approximation $\frac{|V_{SCPKP}|^2}{|A|} \approx \underline{U}$ still holds. ■

B.2 Generation of instances

A set of hundred instances of the SCPKP are generated, with weights and values constructed as follows: For all $j \leq J$, we generate a random value $R_j \in [0,1]$. The value of item (i,j) is $R_j \cdot 350$ plus

a random value in $[-35;35]$. The weight of items (i,j) for every $i \leq I$ is $R_j \cdot 3$ plus a random value in $[-1.5;1.5]$. The capacity C is selected randomly between 0 and the sum of the weights of all items. The idea is to use R_j to create some correlation between the weight, but adding a random value in order to allow for a large enough set of instances. From this set of hundred instances 60 are retained:

B.3 Complete result tables

In **Tables B.1** to **B.6**, we use the following metrics :

- *inst*: the instance number.
- *variant*: experimental variant of CPLEX's B&C considered.
- *C cuts*: number of added CPLEX cuts.
- *U cuts*: number of added UMIC cuts.
- *P cuts*: number of added pattern cuts.
- *P cuts vio*: average violation value of the added pattern cuts.
- *r-value*: linear relaxation value at the root node.
- *r-gap*: gap between the linear relaxation value at the root node and the optimal solution, if optimality is proven by at least one variant.
- *user-time*: proportion of the computational time dedicated to the separation of pattern and UMIC inequalities.
- *#nodes*: number of nodes explored.
- *time/gap*: total computational time in seconds when the instance is solved; gap between the upper and lower bound provided by CPLEX at the time limit when the instance is not solved.

B.4 Types of cuts added by CPLEX

From the sixteen types of cuts available, seven appear in the case of the SCPKP:

1. Cover cuts
2. Generalized Upper Bound Cover cuts (GCover) [107]
3. Flow Cover cuts (FCover)
4. Gomory fractional cuts (Frac) [47]
5. Mixed integer rounding cuts (MIR)
6. Zero-half cuts (ZeroHalf)
7. Lift and project cuts (LiftProj)

Tables B.7 to **B.12** depict for each variant the number of each cuts. The results are averaged for the 10 sets of patterns for variants Cplx+Psep and Cplx+preP. Clearly the majority of the cuts are Cover and MIR ones. For some instances, the number of ZeroHalf cuts can also be significant. For any other cut types, only few cuts are added.

For instances where Cplx is the most efficient variant, we cannot find any difference in the types of cut added compared to other instances.

inst	variant	C cuts	U cuts	P cuts	P cuts vio	r value	r-gap	user time	#nodes	time/gap
	Cplx	345	-	-	-	8125.38	2.66%	-	3625008	353.2
	Umic	-	94	-	-	8205.39	3.67%	1.0%	80545158	2874.6
	Psep	-	-	100	0.48	8087.65	2.18%	38.4%	21483	1.9
1	Cplx+Umic	238	15	-	-	8125.38	2.66%	0.6%	13783767	1144.2
	Cplx+Psep	74	-	100	0.6	8067.12	1.92%	72.6%	3672	1.2
	Psep+Umic	-	18	100	0.44	8087.65	2.18%	63.3%	29865	3.9
	All	74	0	100	0.6	8067.12	1.92%	75.4%	3672	1.2
	Cplx	305	-	-	-	8498.94	3.34%	-	1048228	135.3
	Umic	-	312	-	-	8501.42	3.37%	6.5%	1056401	67.6
	Psep	-	-	100	0.34	8427.76	2.47%	1.2%	270933	19.1
2	Cplx+Umic	235	24	-	-	8496.87	3.31%	0.6%	1027977	102.7
	Cplx+Psep	143	-	100	0.32	8426.69	2.46%	1.1%	208598	19.9
	Psep+Umic	-	0	100	0.34	8427.76	2.47%	1.4%	270933	18.5
	All	143	0	100	0.32	8426.69	2.46%	1.3%	208598	18.4
	Cplx	405	-	-	-	5234.55	9.41%	-	1849854	173.0
	Umic	-	426	-	-	5214.77	8.99%	8.3%	2095765	160.9
	Psep	-	-	7	0.9	4829.68	0.95%	35.5%	0	0.0
3	Cplx+Umic	58	315	-	-	5200.3	8.69%	6.7%	2088692	164.9
	Cplx+Psep	5	-	3	0.76	4795.72	0.24%	17.9%	0	0.0
	Psep+Umic	-	0	7	0.9	4829.68	0.95%	33.1%	0	0.0
	All	5	0	3	0.76	4795.72	0.24%	15.9%	0	0.0
	Cplx	405	-	-	-	6575.88	0.90%	-	5442220	1351.5
	Umic	-	39	-	-	6575.69	0.90%	0.8%	5885435	318.2
	Psep	-	-	100	0.26	6575.87	0.90%	0.7%	4735131	377.6
4	Cplx+Umic	227	40	-	-	6575.2	0.89%	0.4%	5738509	882.8
	Cplx+Psep	143	-	100	0.43	6569.79	0.81%	0.4%	3649391	542.1
	Psep+Umic	-	6	100	0.36	6575.62	0.90%	0.8%	4970221	357.5
	All	142	2	100	0.38	6569.75	0.81%	0.4%	3571861	492.4
	Cplx	405	-	-	-	14288.9	1.10%	-	1816805	280.6
	Umic	-	27	-	-	14302.5	1.20%	1.1%	1180341	56.5
	Psep	-	-	100	0.11	14291.1	1.12%	2.7%	1226118	93.2
5	Cplx+Umic	231	33	-	-	14283.4	1.06%	0.7%	1066178	142.9
	Cplx+Psep	143	-	100	0.11	14291.0	1.12%	1.4%	988415	146.4
	Psep+Umic	-	7	100	0.11	14290.5	1.12%	3.5%	1144164	79.9
	All	139	6	100	0.12	14290.5	1.12%	1.6%	1062315	139.9
	Cplx	400	-	-	-	19711.6	0.95%	-	242702	53.4
	Umic	-	38	-	-	19896.3	1.89%	2.1%	1355505	65.8
	Psep	-	-	59	0.19	19921.7	2.02%	7.1%	1334199	82.7
6	Cplx+Umic	237	18	-	-	19706.9	0.92%	1.0%	250298	42.0
	Cplx+Psep	143	-	100	0.08	19711.6	0.95%	9.0%	213517	36.7
	Psep+Umic	-	38	75	0.19	19896.3	1.89%	7.6%	1308569	84.4
	All	144	15	100	0.12	19706.9	0.92%	11.6%	248436	39.2
	Cplx	405	-	-	-	13801.3	1.78%	-	8463734	1690.0
	Umic	-	345	-	-	13874.4	2.32%	2.2%	6975709	473.7
	Psep	-	-	100	0.39	13821.2	1.92%	0.6%	6770252	541.6
7	Cplx+Umic	206	81	-	-	13794.3	1.73%	0.4%	8454711	1067.6
	Cplx+Psep	143	-	100	0.5	13783.6	1.65%	0.4%	7102323	921.0
	Psep+Umic	-	0	100	0.39	13821.2	1.92%	0.7%	6770252	487.4
	All	143	0	100	0.5	13783.6	1.65%	0.4%	7102323	858.4
	Cplx	305	-	-	-	19621.1	2.02%	-	22632994	0.45%
	Umic	-	438	-	-	19674.1	2.29%	0.9%	19907454	1697.2
	Psep	-	-	100	0.29	19544.5	1.62%	0.5%	1638025	136.2
8	Cplx+Umic	178	15	-	-	19618.0	2.00%	0.6%	31306241	2831.5
	Cplx+Psep	3	-	2	0.74	19232.8	0.00%	17.8%	0	0.0
	Psep+Umic	-	0	100	0.29	19544.5	1.62%	0.7%	1638025	121.4
	All	3	0	2	0.74	19232.8	0.00%	15.4%	0	0.0
	Cplx	305	-	-	-	21935.5	0.54%	-	1759737	248.0
	Umic	-	26	-	-	21940.7	0.57%	1.3%	2189901	100.9
	Psep	-	-	100	0.08	21946.3	0.59%	3.0%	2115733	185.2
9	Cplx+Umic	239	12	-	-	21931.8	0.53%	0.4%	1844651	241.3
	Cplx+Psep	164	-	100	0.03	21935.5	0.54%	2.0%	1879557	274.9
	Psep+Umic	-	26	100	0.06	21940.7	0.57%	3.7%	2175143	175.8
	All	143	9	100	0.06	21931.8	0.53%	2.2%	1876011	246.9
	Cplx	405	-	-	-	22807.8	0.73%	-	1415897	299.3
	Umic	-	243	-	-	22806.8	0.73%	2.3%	1531487	119.2
	Psep	-	-	100	0.24	22796.6	0.68%	0.6%	1108389	111.4
10	Cplx+Umic	228	30	-	-	22803.6	0.71%	0.3%	1294867	204.4
	Cplx+Psep	143	-	100	0.35	22796.5	0.68%	0.4%	1190879	185.3
	Psep+Umic	-	0	100	0.24	22796.6	0.68%	0.7%	1108389	102.1
	All	143	0	100	0.35	22796.5	0.68%	0.5%	1190879	169.0

Table B.1: Resolution of the instances of set 1 of the SCPKP (second phase)

inst	variant	C cuts	U cuts	P cuts	P cuts vio	r value	r-gap	user time	#nodes	time/gap
	Cplx	405	-	-	-	11963.9	1.35%	-	6059004	1393.1
	Umic	-	249	-	-	11947.3	1.21%	1.1%	8346938	618.9
	Psep	-	-	100	0.19	11943.0	1.18%	0.6%	6394528	548.9
11	Cplx+Umic	87	263	-	-	11947.3	1.21%	1.1%	8085378	880.2
	Cplx+Psep	143	-	100	0.17	11938.3	1.14%	0.4%	6552805	854.1
	Psep+Umic	-	0	100	0.19	11943.0	1.18%	0.6%	6394528	499.3
	All	143	0	100	0.17	11938.3	1.14%	0.4%	6552805	789.6
	Cplx	350	-	-	-	9456.27	0.83%	-	529784	93.9
	Umic	-	330	-	-	9494.86	1.24%	5.2%	1163916	78.1
	Psep	-	-	4	0.39	9513.88	1.44%	19.8%	1683434	85.9
12	Cplx+Umic	222	54	-	-	9449.21	0.75%	1.2%	551129	61.8
	Cplx+Psep	143	-	100	0.2	9456.27	0.83%	16.6%	626175	87.1
	Psep+Umic	-	330	0	nan	9494.86	1.24%	24.7%	1163916	87.9
	All	126	51	100	0.23	9449.21	0.75%	22.3%	545385	72.1
	Cplx	405	-	-	-	13686.0	1.09%	-	933876	164.0
	Umic	-	352	-	-	13693.3	1.15%	10.7%	1009570	100.7
	Psep	-	-	100	0.2	13683.6	1.08%	1.0%	755541	68.2
13	Cplx+Umic	75	285	-	-	13679.2	1.04%	10.8%	650342	76.9
	Cplx+Psep	143	-	100	0.19	13682.3	1.07%	0.7%	739312	101.7
	Psep+Umic	-	59	100	0.17	13680.7	1.06%	6.5%	676213	65.2
	All	67	129	100	0.15	13679.1	1.04%	5.7%	671996	79.3
	Cplx	210	-	-	-	2164.88	1.14%	-	127247	8.3
	Umic	-	188	-	-	2181.78	1.93%	2.3%	2131653	79.3
	Psep	-	-	100	0.25	2185.22	2.09%	1.5%	2209549	86.5
14	Cplx+Umic	156	45	-	-	2164.22	1.11%	6.6%	202893	17.1
	Cplx+Psep	83	-	100	0.46	2166.64	1.23%	3.2%	224382	17.2
	Psep+Umic	-	188	100	0.24	2181.78	1.93%	2.8%	1807699	79.2
	All	57	42	100	0.55	2164.22	1.11%	9.1%	247563	17.4
	Cplx	305	-	-	-	16614.7	1.56%	-	18942824	2156.6
	Umic	-	14	-	-	16674.8	1.93%	1.5%	26634131	817.4
	Psep	-	-	0	nan	16677.4	1.94%	2.2%	27328495	809.0
15	Cplx+Umic	179	15	-	-	16613.6	1.55%	0.5%	29751941	2579.6
	Cplx+Psep	179	-	4	0.13	16614.7	1.56%	0.7%	30431217	2662.4
	Psep+Umic	-	14	0	nan	16674.8	1.93%	2.4%	26634131	734.8
	All	180	12	0	nan	16613.6	1.55%	0.8%	30693136	2440.7
	Cplx	118	-	-	-	10145.9	0.47%	-	10439	1.1
	Umic	-	20	-	-	10168.0	0.69%	1.0%	4304481	154.9
	Psep	-	-	100	0.36	10168.0	0.69%	0.9%	4005799	156.5
16	Cplx+Umic	234	27	-	-	10144.3	0.46%	1.1%	360889	36.4
	Cplx+Psep	143	-	100	0.29	10145.3	0.47%	0.6%	472577	39.0
	Psep+Umic	-	0	100	0.36	10168.0	0.69%	1.0%	4005799	142.6
	All	143	0	100	0.29	10145.3	0.47%	0.8%	472577	36.6
	Cplx	320	-	-	-	7739.68	1.55%	-	2277828	236.3
	Umic	-	30	-	-	7794.65	2.27%	1.9%	2053024	106.4
	Psep	-	-	100	0.28	7796.51	2.29%	1.4%	1975443	131.8
17	Cplx+Umic	238	15	-	-	7739.39	1.54%	0.7%	2440104	222.1
	Cplx+Psep	143	-	100	0.34	7738.64	1.53%	1.5%	1547491	135.6
	Psep+Umic	-	3	100	0.36	7794.69	2.27%	2.6%	1793807	116.0
	All	143	0	100	0.34	7738.64	1.53%	1.7%	1547491	124.6
	Cplx	405	-	-	-	14608.2	-	-	19599594	0.42%
	Umic	-	18	-	-	14611.6	-	0.8%	58876400	0.36%
	Psep	-	-	100	0.33	14610.7	-	0.7%	48938164	0.38%
18	Cplx+Umic	238	15	-	-	14608.1	-	0.4%	28426400	0.41%
	Cplx+Psep	143	-	100	0.38	14608.1	-	0.5%	35613800	0.4 %
	Psep+Umic	-	0	100	0.33	14610.7	-	0.8%	53262133	0.37%
	All	143	0	100	0.38	14608.1	-	0.5%	34065400	0.4 %
	Cplx	400	-	-	-	1486.25	1.71%	-	103512	21.0
	Umic	-	288	-	-	1490.72	2.01%	4.3%	1565566	108.6
	Psep	-	-	100	0.23	1479.22	1.23%	3.2%	131487	14.3
19	Cplx+Umic	221	45	-	-	1485.78	1.68%	2.2%	250723	38.0
	Cplx+Psep	143	-	100	0.22	1473.75	0.85%	5.7%	53333	8.4
	Psep+Umic	-	0	100	0.23	1479.22	1.23%	3.5%	131487	12.6
	All	143	0	100	0.22	1473.75	0.85%	6.5%	53333	8.6
	Cplx	405	-	-	-	27827.6	0.87%	-	1453934	325.0
	Umic	-	149	-	-	27845.0	0.93%	1.8%	1784950	107.0
	Psep	-	-	100	0.26	27819.0	0.83%	0.7%	702933	69.5
20	Cplx+Umic	238	12	-	-	27838.6	0.91%	0.3%	1562808	257.6
	Cplx+Psep	143	-	100	0.36	27819.0	0.83%	0.5%	567484	86.2
	Psep+Umic	-	0	100	0.26	27819.0	0.83%	0.8%	702933	62.0
	All	143	0	100	0.36	27819.0	0.83%	0.5%	567484	89.2

Table B.2: Resolution of the instances of set 1 of the SCPKP (second phase)

inst	variant	C cuts	U cuts	P cuts	P cuts vio	r value	r-gap	user time	#nodes	time/gap
	Cplx	605	-	-	-	6672.83	1.14%	-	10830484	0.08%
	Umic	-	482	-	-	6675.75	1.18%	1.2%	18347628	1387.1
	Psep	-	-	100	0.21	6643.91	0.70%	57.6%	1381	1.0
21	Cplx+Umic	354	25	-	-	6672.14	1.13%	0.4%	17067043	2390.8
	Cplx+Psep	4	-	2	0.48	6597.65	0.00%	43.2%	0	0.0
	Psep+Umic	-	0	100	0.21	6643.91	0.70%	71.2%	1381	1.2
	All	4	0	2	0.48	6597.65	0.00%	41.9%	0	0.0
	Cplx	605	-	-	-	42275.2	0.35%	-	1791840	871.6
	Umic	-	306	-	-	42290.1	0.39%	2.5%	1722333	172.2
	Psep	-	-	100	0.33	42301.2	0.41%	0.6%	1133096	116.0
22	Cplx+Umic	342	61	-	-	42271.1	0.34%	0.2%	1974608	678.0
	Cplx+Psep	263	-	100	0.06	42273.2	0.35%	0.2%	1052428	339.6
	Psep+Umic	-	0	100	0.33	42301.2	0.41%	0.7%	1133096	104.2
	All	262	3	100	0.07	42271.0	0.34%	0.3%	1097905	331.4
	Cplx	227	-	-	-	7961.65	1.33%	-	31200	4.4
	Umic	-	291	-	-	7962.61	1.35%	1.8%	6100604	479.0
	Psep	-	-	100	0.21	8051.68	2.48%	0.5%	25484185	2344.5
23	Cplx+Umic	360	9	-	-	7961.65	1.33%	0.2%	3351591	652.8
	Cplx+Psep	263	-	100	0.26	7957.28	1.28%	0.4%	3523210	665.0
	Psep+Umic	-	291	100	0.21	7962.61	1.35%	1.5%	5231512	571.1
	All	261	6	100	0.24	7961.65	1.33%	0.5%	3531747	596.7
	Cplx	218	-	-	-	28987.0	0.25%	-	67547	9.8
	Umic	-	9	-	-	29027.2	0.39%	0.9%	4945832	235.2
	Psep	-	-	0	nan	29030.7	0.40%	4.9%	4911185	245.0
24	Cplx+Umic	358	12	-	-	28987.0	0.25%	0.3%	1960764	408.0
	Cplx+Psep	363	-	0	nan	28987.0	0.25%	2.5%	2256372	463.9
	Psep+Umic	-	9	0	nan	29027.2	0.39%	5.6%	4945832	220.2
	All	360	9	0	nan	28987.4	0.25%	2.7%	2494889	465.7
	Cplx	605	-	-	-	28438.1	0.66%	-	12354812	0.16%
	Umic	-	6	-	-	28455.7	0.72%	0.9%	28764074	1537.8
	Psep	-	-	0	nan	28456.6	0.73%	1.8%	28518001	1509.9
25	Cplx+Umic	360	9	-	-	28438.1	0.66%	0.2%	14436657	0.2 %
	Cplx+Psep	363	-	0	nan	28438.1	0.66%	0.6%	14007958	0.2 %
	Psep+Umic	-	6	0	nan	28455.7	0.72%	2.1%	28764074	1412.8
	All	361	6	0	nan	28438.1	0.66%	0.7%	14962100	0.19%
	Cplx	5	-	-	-	13369.2	0.29%	-	1507	0.2
	Umic	-	368	-	-	13377.6	0.35%	75.8%	24847	11.6
	Psep	-	-	1	0.98	13385.3	0.41%	92.3%	23136	18.7
26	Cplx+Umic	194	102	-	-	13363.2	0.24%	59.7%	4324	1.7
	Cplx+Psep	237	-	10	0.1	13369.2	0.29%	94.0%	5115	16.4
	Psep+Umic	-	66	1	0.98	13377.4	0.35%	92.6%	23300	20.6
	All	176	123	10	0.12	13363.2	0.24%	96.1%	4771	18.7
	Cplx	605	-	-	-	8548.64	1.17%	-	2834093	778.5
	Umic	-	390	-	-	8544.39	1.12%	2.0%	6337030	618.0
	Psep	-	-	14	0.43	8453.1	0.04%	59.8%	3	0.1
27	Cplx+Umic	228	218	-	-	8544.35	1.12%	0.8%	4590923	820.1
	Cplx+Psep	2	-	5	0.41	8454.58	0.06%	34.3%	0	0.0
	Psep+Umic	-	0	14	0.43	8453.1	0.04%	89.6%	3	0.2
	All	2	0	5	0.41	8454.58	0.06%	34.6%	0	0.0
	Cplx	605	-	-	-	9552.2	0.38%	-	5278948	2040.7
	Umic	-	9	-	-	9577.34	0.65%	1.0%	4911215	260.5
	Psep	-	-	30	0.91	9562.54	0.49%	2.7%	5299676	450.9
28	Cplx+Umic	356	18	-	-	9551.93	0.38%	0.2%	4876800	1175.8
	Cplx+Psep	268	-	95	0.18	9549.92	0.36%	2.5%	1573932	430.4
	Psep+Umic	-	3	37	0.63	9562.5	0.49%	3.3%	4748985	377.5
	All	259	12	100	0.27	9549.69	0.36%	2.0%	2078450	507.8
	Cplx	605	-	-	-	12865.2	1.07%	-	9946516	0.28%
	Umic	-	696	-	-	12914.1	1.46%	1.1%	26036123	0.47%
	Psep	-	-	100	0.37	12870.7	1.11%	0.4%	14287036	1691.6
29	Cplx+Umic	335	72	-	-	12860.9	1.04%	0.3%	18331800	0.41%
	Cplx+Psep	263	-	100	0.29	12825.7	0.76%	0.2%	8591393	1876.6
	Psep+Umic	-	0	100	0.37	12870.7	1.11%	0.4%	14287036	1542.3
	All	259	12	100	0.29	12825.2	0.76%	0.3%	7896841	1656.6
	Cplx	5	-	-	-	36394.2	0.25%	-	6657	0.6
	Umic	-	462	-	-	36424.9	0.33%	13.0%	507285	61.2
	Psep	-	-	1	0.3	36444.2	0.39%	16.2%	626080	36.9
30	Cplx+Umic	350	39	-	-	36392.1	0.24%	0.8%	121340	38.1
	Cplx+Psep	362	-	1	0.3	36394.2	0.25%	14.2%	117803	42.4
	Psep+Umic	-	231	1	0.3	36426.7	0.34%	28.5%	435076	44.4
	All	346	48	1	0.3	36392.1	0.24%	15.5%	123822	42.0

Table B.3: Resolution of the instances of set 2 of the SCPKP (second phase)

inst	variant	C cuts	U cuts	P cuts	P cuts vio	r value	r-gap	user time	#nodes	time/gap
	Cplx	605	-	-	-	8611.38	1.00%	-	7909784	0.31%
	Umic	-	333	-	-	8610.7	0.99%	1.4%	24628963	1655.1
	Psep	-	-	100	0.15	8610.37	0.99%	0.8%	18721035	1415.6
31	Cplx+Umic	322	84	-	-	8609.17	0.98%	0.3%	17050294	0.18%
	Cplx+Psep	263	-	100	0.11	8608.34	0.97%	0.2%	16262900	0.01%
	Psep+Umic	-	0	100	0.15	8610.37	0.99%	0.8%	18721035	1268.0
	All	263	0	100	0.11	8608.34	0.97%	0.3%	16364640	3118.8
	Cplx	415	-	-	-	39078.4	0.27%	-	7410282	1375.8
	Umic	-	6	-	-	39171.6	0.51%	1.1%	17142128	685.9
	Psep	-	-	1	0.85	39079.8	0.28%	4.9%	4162387	120.4
32	Cplx+Umic	359	12	-	-	39077.1	0.27%	0.3%	6547609	1054.3
	Cplx+Psep	6	-	1	0.85	38998.1	0.07%	16.5%	0	0.0
	Psep+Umic	-	12	1	0.85	39077.3	0.27%	5.6%	4455866	114.0
	All	8	8	1	0.85	38998.1	0.07%	82.5%	0	0.1
	Cplx	27	-	-	-	52298.5	0.35%	-	944	0.1
	Umic	-	27	-	-	52298.8	0.35%	1.4%	4690029	166.3
	Psep	-	-	100	0.19	52299.2	0.35%	1.9%	5198175	254.3
33	Cplx+Umic	32	24	-	-	52288.2	0.33%	82.7%	491	0.3
	Cplx+Psep	37	-	6	0.19	52298.5	0.35%	72.1%	639	0.2
	Psep+Umic	-	27	100	0.19	52298.8	0.35%	2.4%	4359231	181.6
	All	36	18	17	0.05	52288.2	0.33%	90.1%	491	0.6
	Cplx	210	-	-	-	30600.1	0.48%	-	607637	52.9
	Umic	-	9	-	-	30604.9	0.49%	1.6%	1813800	60.3
	Psep	-	-	2	0.72	30590.9	0.45%	27.5%	643757	35.9
34	Cplx+Umic	360	9	-	-	30600.1	0.48%	0.3%	1306558	213.1
	Cplx+Psep	2	-	1	0.76	30454.9	0.00%	31.0%	0	0.0
	Psep+Umic	-	9	1	0.76	30589.6	0.44%	31.3%	624509	33.4
	All	2	0	1	0.76	30454.9	0.00%	31.1%	0	0.0
	Cplx	7	-	-	-	21810.5	0.00%	-	0	0.0
	Umic	-	201	-	-	21828.4	0.08%	97.3%	0	1.5
	Psep	-	-	10	0.02	22046.9	1.08%	7.7%	1591246	111.4
35	Cplx+Umic	7	6	-	-	21810.5	0.00%	46.9%	0	0.0
	Cplx+Psep	7	-	0	nan	21810.5	0.00%	22.8%	0	0.0
	Psep+Umic	-	201	0	nan	21828.4	0.08%	97.0%	0	1.3
	All	7	3	0	nan	21810.5	0.00%	48.1%	0	0.0
	Cplx	137	-	-	-	43499.4	0.35%	-	111433	8.3
	Umic	-	22	-	-	43500.2	0.35%	1.1%	11442996	405.4
	Psep	-	-	100	0.35	43496.7	0.34%	0.6%	2813931	195.8
36	Cplx+Umic	354	18	-	-	43499.2	0.35%	0.4%	435114	86.1
	Cplx+Psep	62	-	100	0.57	43426.3	0.18%	11.8%	4997	0.4
	Psep+Umic	-	6	100	0.36	43496.7	0.34%	1.1%	3251727	207.9
	All	62	0	100	0.57	43426.3	0.18%	23.5%	4997	0.5
	Cplx	235	-	-	-	19087.5	0.83%	-	2542818	273.4
	Umic	-	12	-	-	19087.5	0.83%	1.3%	5910018	191.9
	Psep	-	-	100	0.37	19090.1	0.85%	1.1%	6229189	308.8
37	Cplx+Umic	135	18	-	-	19087.5	0.83%	0.6%	1881443	149.8
	Cplx+Psep	41	-	100	0.35	19087.5	0.83%	0.8%	5585807	329.7
	Psep+Umic	-	12	100	0.44	19088.8	0.84%	1.2%	6672052	297.9
	All	29	28	100	0.25	19087.5	0.83%	1.1%	3979765	233.2
	Cplx	455	-	-	-	19670.9	0.36%	-	3138775	832.5
	Umic	-	12	-	-	19674.7	0.38%	0.8%	3101363	170.6
	Psep	-	-	100	0.17	19675.3	0.39%	0.5%	3238464	380.9
38	Cplx+Umic	358	12	-	-	19670.4	0.36%	0.2%	3201752	896.5
	Cplx+Psep	263	-	100	0.24	19670.7	0.36%	0.2%	3293933	809.0
	Psep+Umic	-	9	100	0.18	19674.7	0.38%	0.5%	3201700	335.7
	All	263	0	100	0.24	19670.7	0.36%	0.2%	3293933	855.2
	Cplx	605	-	-	-	19774.0	0.74%	-	12104994	0.19%
	Umic	-	30	-	-	19774.4	0.75%	0.8%	45650680	2538.8
	Psep	-	-	100	0.51	19772.5	0.74%	0.5%	35536030	0.07%
39	Cplx+Umic	351	30	-	-	19773.3	0.74%	0.2%	18092893	0.14%
	Cplx+Psep	263	-	100	0.48	19772.5	0.74%	0.2%	19016360	0.17%
	Psep+Umic	-	0	100	0.51	19772.5	0.74%	0.5%	39402500	0.06%
	All	263	0	100	0.48	19772.5	0.74%	0.3%	18567926	0.17%
	Cplx	605	-	-	-	38485.6	0.36%	-	10125494	0.11%
	Umic	-	21	-	-	38496.8	0.39%	0.9%	70754543	3151.5
	Psep	-	-	100	0.34	38496.8	0.39%	0.8%	51413500	0.04%
40	Cplx+Umic	352	18	-	-	38484.5	0.36%	0.2%	15052400	0.11%
	Cplx+Psep	263	-	100	0.29	38484.1	0.35%	0.3%	19122358	0.09%
	Psep+Umic	-	0	100	0.34	38496.8	0.39%	0.8%	56723700	0.03%
	All	263	0	100	0.29	38484.1	0.35%	0.3%	18554100	0.1 %

Table B.4: Resolution of the instances of set 2 of the SCPKP (second phase)

inst	variant	C cuts	U cuts	P cuts	P cuts vio	r value	r-gap	user time	#nodes	time/gap
	Cplx	905	-	-	-	39387.8	0.56%	-	1147464	805.1
	Umic	-	108	-	-	39392.1	0.57%	1.2%	988599	92.3
	Psep	-	-	100	0.18	39395.3	0.58%	0.5%	1073907	138.9
41	Cplx+Umic	536	21	-	-	39376.9	0.53%	0.1%	1007053	619.0
	Cplx+Psep	443	-	100	0.17	39268.5	0.25%	1.3%	31038	16.6
	Psep+Umic	-	18	100	0.29	39381.2	0.54%	0.9%	1006298	118.7
	All	432	24	100	0.19	39267.1	0.25%	2.6%	30758	15.2
	Cplx	770	-	-	-	43984.0	0.61%	-	3693204	2452.6
	Umic	-	101	-	-	43984.2	0.61%	1.2%	3358215	250.5
	Psep	-	-	15	0.08	44062.4	0.79%	2.0%	4544901	292.8
42	Cplx+Umic	534	18	-	-	43973.6	0.58%	0.1%	3203088	1441.7
	Cplx+Psep	443	-	100	0.07	43984.0	0.61%	0.3%	3185171	1208.5
	Psep+Umic	-	101	7	0.08	43984.2	0.61%	2.7%	3833602	263.0
	All	437	15	100	0.07	43973.6	0.58%	0.3%	3001514	1101.1
	Cplx	605	-	-	-	35842.2	0.30%	-	2358605	725.3
	Umic	-	9	-	-	35855.1	0.34%	0.6%	2289708	164.6
	Psep	-	-	0	nan	35856.8	0.34%	3.8%	2173793	152.1
43	Cplx+Umic	360	9	-	-	35841.5	0.30%	0.2%	2062250	613.7
	Cplx+Psep	363	-	0	nan	35842.2	0.30%	1.0%	2086498	606.1
	Psep+Umic	-	9	0	nan	35856.1	0.34%	4.0%	2287786	143.7
	All	361	6	0	nan	35841.5	0.30%	1.2%	1983926	547.7
	Cplx	560	-	-	-	27403.7	0.49%	-	11954214	0.09%
	Umic	-	12	-	-	27409.7	0.51%	0.8%	15852557	935.6
	Psep	-	-	100	0.19	27411.0	0.51%	0.7%	15539237	1120.6
44	Cplx+Umic	362	21	-	-	27403.7	0.49%	0.2%	15865687	0.07%
	Cplx+Psep	263	-	100	0.14	27403.7	0.49%	0.2%	17989260	0.05%
	Psep+Umic	-	12	100	0.17	27409.8	0.51%	0.8%	16270349	1049.5
	All	259	12	100	0.11	27403.7	0.49%	0.3%	19431792	3546.4
	Cplx	519	-	-	-	41001.8	0.56%	-	760888	279.2
	Umic	-	26	-	-	41001.5	0.56%	0.8%	877880	59.3
	Psep	-	-	100	0.11	41002.8	0.56%	0.6%	973276	83.9
45	Cplx+Umic	538	12	-	-	41001.1	0.56%	0.1%	855127	497.3
	Cplx+Psep	443	-	100	0.13	41001.8	0.56%	0.1%	914876	452.1
	Psep+Umic	-	21	100	0.08	41001.5	0.56%	0.7%	939816	75.3
	All	442	3	100	0.06	41001.1	0.56%	0.1%	902994	404.4
	Cplx	705	-	-	-	19874.4	0.82%	-	1633243	528.2
	Umic	-	335	-	-	19837.3	0.63%	19.4%	1395433	171.9
	Psep	-	-	14	0.22	19841.7	0.65%	8.5%	917318	74.2
46	Cplx+Umic	320	267	-	-	19840.8	0.65%	6.1%	1650862	425.1
	Cplx+Psep	7	-	2	1.29	19730.3	0.09%	14.4%	0	0.0
	Psep+Umic	-	6	9	0.3	19840.8	0.65%	9.4%	934128	70.0
	All	7	3	1	1.57	19730.3	0.09%	88.3%	0	0.1
	Cplx	745	-	-	-	19858.7	0.76%	-	1042047	299.5
	Umic	-	36	-	-	19867.0	0.80%	1.6%	1686472	102.6
	Psep	-	-	0	nan	19870.6	0.82%	9.5%	1690735	105.8
47	Cplx+Umic	535	21	-	-	19853.2	0.73%	0.4%	1609470	383.8
	Cplx+Psep	543	-	0	nan	19858.7	0.76%	2.6%	1651034	412.1
	Psep+Umic	-	36	0	nan	19867.0	0.80%	10.9%	1689543	101.2
	All	535	18	1	0.38	19853.2	0.73%	3.3%	1693933	371.2
	Cplx	475	-	-	-	2311.63	2.08%	-	13556897	2597.2
	Umic	-	183	-	-	2322.33	2.55%	0.9%	13346766	867.4
	Psep	-	-	39	0.23	2323.39	2.60%	3.0%	12000229	738.3
48	Cplx+Umic	293	16	-	-	2311.63	2.08%	0.3%	12412538	2026.0
	Cplx+Psep	200	-	100	0.16	2311.63	2.08%	0.9%	12719732	1789.5
	Psep+Umic	-	183	100	0.22	2322.68	2.57%	2.2%	12753438	1004.9
	All	194	13	100	0.17	2311.63	2.08%	0.7%	12334318	1638.7
	Cplx	905	-	-	-	2274.55	3.33%	-	840705	359.1
	Umic	-	1044	-	-	2272.52	3.23%	31.6%	828917	201.3
	Psep	-	-	100	0.47	2220.74	0.88%	79.8%	45	0.6
49	Cplx+Umic	140	633	-	-	2270.72	3.15%	16.1%	882489	217.6
	Cplx+Psep	1	-	100	0.36	2220.28	0.86%	76.8%	125	0.8
	Psep+Umic	-	0	100	0.47	2220.74	0.88%	83.9%	45	0.6
	All	1	0	100	0.36	2220.28	0.86%	81.0%	125	0.8
	Cplx	630	-	-	-	34994.5	0.54%	-	10303438	0.23%
	Umic	-	12	-	-	35039.6	0.67%	0.8%	41293662	2296.7
	Psep	-	-	0	nan	35041.0	0.67%	0.9%	42987493	2514.4
50	Cplx+Umic	374	12	-	-	34993.7	0.54%	0.2%	14044700	0.21%
	Cplx+Psep	378	-	0	nan	34994.5	0.54%	0.3%	13951413	0.2 %
	Psep+Umic	-	12	0	nan	35039.6	0.67%	1.1%	35768862	1778.2
	All	375	9	0	nan	34993.7	0.54%	0.3%	14563900	0.2 %

Table B.5: Resolution of the instances of set 3 of the SCPKP (second phase)

inst	variant	C cuts	U cuts	P cuts	P cuts vio	r value	r-gap	user time	#nodes	time/gap
	Cplx	615	-	-	-	31853.1	0.58%	-	188336	65.6
	Umic	-	9	-	-	31902.5	0.74%	1.1%	493406	40.9
	Psep	-	-	0	nan	31905.2	0.75%	14.1%	574701	49.4
51	Cplx+Umic	375	18	-	-	31852.4	0.58%	1.6%	84143	24.8
	Cplx+Psep	381	-	0	nan	31853.1	0.58%	19.2%	98648	37.0
	Psep+Umic	-	9	0	nan	31902.5	0.74%	17.1%	492657	42.2
	All	377	12	0	nan	31852.4	0.58%	21.5%	98288	34.6
	Cplx	905	-	-	-	23664.4	0.49%	-	1730646	930.5
	Umic	-	192	-	-	23716.9	0.72%	3.8%	1499676	128.6
	Psep	-	-	0	nan	23763.8	0.92%	6.4%	1450565	89.2
52	Cplx+Umic	453	207	-	-	23658.6	0.47%	1.2%	1283791	470.7
	Cplx+Psep	543	-	0	nan	23664.4	0.49%	0.8%	1712424	771.4
	Psep+Umic	-	192	0	nan	23716.9	0.72%	8.2%	1499676	120.8
	All	522	63	0	nan	23660.6	0.48%	1.4%	1234787	487.1
	Cplx	820	-	-	-	29884.3	0.45%	-	903545	559.2
	Umic	-	9	-	-	29902.1	0.51%	0.6%	739761	57.3
	Psep	-	-	0	nan	29911.9	0.55%	6.0%	706313	60.5
53	Cplx+Umic	540	9	-	-	29886.0	0.46%	0.1%	774691	401.6
	Cplx+Psep	543	-	0	nan	29884.3	0.45%	1.0%	697532	390.6
	Psep+Umic	-	9	0	nan	29902.1	0.51%	6.9%	719822	53.4
	All	540	9	0	nan	29878.2	0.43%	1.4%	570611	310.4
	Cplx	305	-	-	-	5794.38	1.80%	-	2129326	294.3
	Umic	-	225	-	-	5808.69	2.05%	3.0%	1720151	93.7
	Psep	-	-	100	0.24	5808.71	2.05%	1.3%	1680084	102.4
54	Cplx+Umic	190	69	-	-	5803.29	1.95%	0.6%	1619294	181.4
	Cplx+Psep	116	-	100	0.49	5798.79	1.87%	0.9%	1495314	154.7
	Psep+Umic	-	180	100	0.29	5808.71	2.05%	3.2%	1465986	98.9
	All	112	6	100	0.43	5798.79	1.87%	1.1%	1374869	128.7
	Cplx	905	-	-	-	12667.1	0.95%	-	746964	483.2
	Umic	-	941	-	-	12630.8	0.66%	99.2%	0	108.2
	Psep	-	-	0	nan	12726.9	1.43%	4.2%	3570429	236.4
55	Cplx+Umic	540	9	-	-	12666.9	0.95%	0.1%	1036185	444.5
	Cplx+Psep	543	-	0	nan	12667.1	0.95%	3.0%	667293	306.0
	Psep+Umic	-	941	0	nan	12630.8	0.66%	99.2%	0	94.2
	All	541	6	0	nan	12666.9	0.95%	2.3%	1007083	411.0
	Cplx	631	-	-	-	3789.71	3.51%	-	54765	16.8
	Umic	-	471	-	-	3762.51	2.77%	10.2%	1272533	178.7
	Psep	-	-	100	0.47	3757.03	2.62%	0.4%	990985	215.8
56	Cplx+Umic	1	758	-	-	3761.98	2.75%	13.5%	1101652	210.5
	Cplx+Psep	64	-	100	0.42	3711.58	1.38%	50.0%	1283	0.8
	Psep+Umic	-	0	100	0.47	3757.03	2.62%	0.5%	990985	191.4
	All	64	0	100	0.42	3711.58	1.38%	55.5%	1283	1.0
	Cplx	23	-	-	-	54958.6	0.51%	-	281286	27.6
	Umic	-	549	-	-	54958.7	0.51%	2.7%	5592133	1026.0
	Psep	-	-	100	0.38	55197.4	0.95%	0.4%	5387363	569.4
57	Cplx+Umic	451	72	-	-	54950.9	0.49%	0.1%	5515744	2474.1
	Cplx+Psep	481	-	2	0.56	54958.6	0.51%	0.2%	5314998	2293.3
	Psep+Umic	-	226	100	0.22	54955.9	0.50%	1.6%	6658806	993.1
	All	461	51	2	0.56	54953.3	0.50%	0.3%	5223371	2306.2
	Cplx	855	-	-	-	33248.3	0.31%	-	1079727	525.5
	Umic	-	9	-	-	33284.3	0.42%	0.6%	4258811	267.7
	Psep	-	-	0	nan	33289.9	0.43%	1.6%	4622787	324.3
58	Cplx+Umic	513	49	-	-	33247.0	0.30%	0.3%	988586	532.9
	Cplx+Psep	543	-	0	nan	33248.3	0.31%	0.8%	994129	499.6
	Psep+Umic	-	9	0	nan	33284.3	0.42%	1.9%	4309053	278.8
	All	514	49	0	nan	33247.0	0.30%	1.2%	977746	504.5
	Cplx	463	-	-	-	14145.4	0.97%	-	204678	47.1
	Umic	-	471	-	-	14149.3	1.00%	31.9%	538179	101.4
	Psep	-	-	100	0.21	14121.8	0.80%	0.9%	64250	14.7
59	Cplx+Umic	474	25	-	-	14144.8	0.97%	0.2%	250562	103.2
	Cplx+Psep	383	-	100	0.22	14119.5	0.79%	0.5%	55189	26.1
	Psep+Umic	-	0	100	0.21	14121.8	0.80%	2.5%	64250	13.1
	All	383	0	100	0.22	14119.5	0.79%	1.1%	55189	27.6
	Cplx	565	-	-	-	17107.7	2.05%	-	1365449	361.2
	Umic	-	282	-	-	17105.0	2.03%	3.0%	1119149	87.5
	Psep	-	-	100	0.24	17096.8	1.98%	0.5%	724865	85.5
60	Cplx+Umic	374	123	-	-	17097.5	1.99%	0.5%	1149104	245.2
	Cplx+Psep	323	-	100	0.3	17076.3	1.86%	0.3%	642746	138.2
	Psep+Umic	-	49	100	0.15	17080.3	1.89%	1.2%	777701	84.0
	All	320	6	100	0.29	17071.7	1.83%	0.3%	668944	155.5

Table B.6: Resolution of the instances of set 3 of the SCPKP (second phase)

inst	variant	Cover	GCover	FCover	Frac	Mir	ZeroHalf	LiftProj
1	Cplx	212	0	0	3	127	3	0
	Cplx+Umic	235	1	0	1	127	1	0
	Cplx+Psep	72	1	0	1	127	1	0
	All	72	1	0	1	127	1	0
2	Cplx	239	3	0	6	56	1	0
	Cplx+Umic	229	3	0	2	4	1	0
	Cplx+Psep	139	3	0	1	1	2	0
	All	139	3	0	1	1	2	0
3	Cplx	340	1	0	5	50	9	0
	Cplx+Umic	55	1	0	1	1	1	0
	Cplx+Psep	55	1	1	2	1	2	0
	All	55	1	1	2	1	2	0
4	Cplx	159	0	0	3	239	4	0
	Cplx+Umic	225	0	0	2	239	4	0
	Cplx+Psep	141	0	0	1	1	4	0
	All	141	0	0	1	1	4	0
5	Cplx	178	0	0	4	223	0	0
	Cplx+Umic	227	0	0	2	1	1	0
	Cplx+Psep	141	0	0	1	1	1	1
	All	137	0	0	2	1	1	1
6	Cplx	2	0	0	5	595	3	0
	Cplx+Umic	349	0	0	1	595	4	0
	Cplx+Psep	349	1	1	2	595	4	0
	All	349	1	1	2	595	4	0
7	Cplx	567	0	0	5	26	7	0
	Cplx+Umic	338	0	3	1	26	7	0
	Cplx+Psep	259	0	2	1	1	7	0
	All	260	0	1	1	1	7	0
8	Cplx	219	0	0	4	3	1	0
	Cplx+Umic	355	0	1	2	1	1	0
	Cplx+Psep	257	0	1	2	2	1	1
	All	256	0	1	2	2	1	1
9	Cplx	176	0	0	3	39	0	0
	Cplx+Umic	352	0	0	2	3	1	0
	Cplx+Psep	357	0	0	2	2	2	0
	All	355	0	0	2	3	2	0
10	Cplx	307	0	0	4	218	76	0
	Cplx+Umic	359	0	0	1	218	76	0
	Cplx+Psep	361	0	0	1	1	76	0
	All	360	0	0	1	1	76	0

Table B.7: Number of each type of cuts added by CPLEX for each variant for instances 1 to 10

inst	variant	Cover	GCover	FCover	Frac	Mir	ZeroHalf	LiftProj
11	Cplx	886	0	0	4	11	4	0
	Cplx+Umic	534	0	0	2	11	4	0
	Cplx+Psep	437	0	0	2	1	3	0
	All	429	0	0	2	1	1	0
12	Cplx	92	0	0	4	642	32	0
	Cplx+Umic	533	0	0	1	642	32	0
	Cplx+Psep	440	1	0	1	642	1	0
	All	436	1	0	1	642	1	0
13	Cplx	577	0	0	3	25	0	0
	Cplx+Umic	357	0	0	1	2	0	0
	Cplx+Psep	359	0	0	1	2	0	1
	All	358	0	0	1	2	0	1
14	Cplx	479	0	0	3	76	2	0
	Cplx+Umic	359	0	0	3	76	2	0
	Cplx+Psep	259	0	0	3	1	2	0
	All	256	0	0	3	1	2	0
15	Cplx	430	0	0	2	86	1	0
	Cplx+Umic	537	0	0	1	86	1	0
	Cplx+Psep	441	0	0	1	86	1	1
	All	441	0	0	1	86	1	1
16	Cplx	385	0	0	6	8	1	0
	Cplx+Umic	228	0	0	2	5	2	0
	Cplx+Psep	135	0	0	3	4	1	0
	All	137	0	0	1	4	2	0
17	Cplx	221	0	0	5	179	0	0
	Cplx+Umic	204	0	0	1	1	0	0
	Cplx+Psep	142	0	0	1	1	0	0
	All	142	0	0	1	1	0	0
18	Cplx	219	0	0	4	80	2	0
	Cplx+Umic	177	0	0	1	80	2	0
	Cplx+Psep	177	0	0	2	80	2	1
	All	177	0	0	2	80	2	1
19	Cplx	290	3	0	5	6	1	0
	Cplx+Umic	238	3	0	1	6	1	0
	Cplx+Psep	159	1	1	1	1	1	0
	All	142	1	1	1	1	1	0
20	Cplx	362	0	0	4	38	1	0
	Cplx+Umic	225	0	0	2	1	1	0
	Cplx+Psep	140	0	0	2	1	1	0
	All	140	0	0	2	1	1	0

Table B.8: Number of each type of cuts added by CPLEX for each variant for instances 11 to 20

inst	variant	Cover	GCover	FCover	Frac	Mir	ZeroHalf	LiftProj
21	Cplx	1	0	0	4	0	0	0
	Cplx+Umic	191	0	0	1	2	0	0
	Cplx+Psep	233	2	0	1	1	0	0
	All	171	2	0	1	3	1	0
22	Cplx	586	0	0	5	13	1	0
	Cplx+Umic	225	0	1	2	13	1	0
	Cplx+Psep	225	0	1	1	1	1	0
	All	225	0	1	1	1	1	0
23	Cplx	138	0	0	6	439	22	0
	Cplx+Umic	352	0	0	2	2	22	0
	Cplx+Psep	265	0	0	2	1	22	0
	All	256	0	0	2	1	22	0
24	Cplx	46	0	0	4	550	5	0
	Cplx+Umic	333	0	0	2	550	5	0
	Cplx+Psep	258	0	0	1	550	4	0
	All	257	0	0	1	550	1	0
25	Cplx	1	0	0	3	1	0	0
	Cplx+Umic	348	0	0	2	1	0	0
	Cplx+Psep	360	0	0	2	1	0	0
	All	344	0	0	2	1	0	0
26	Cplx	690	0	0	4	8	3	0
	Cplx+Umic	319	0	0	1	8	3	0
	Cplx+Psep	319	0	1	3	8	1	2
	All	319	0	1	3	8	1	2
27	Cplx	719	0	0	2	23	1	0
	Cplx+Umic	530	0	0	2	1	2	0
	Cplx+Psep	539	0	0	2	1	2	1
	All	530	0	0	2	1	2	1
28	Cplx	349	1	0	2	115	8	0
	Cplx+Umic	286	1	0	3	3	8	0
	Cplx+Psep	195	1	0	2	2	1	0
	All	188	1	0	3	3	1	0
29	Cplx	827	1	0	6	69	2	0
	Cplx+Umic	137	1	0	2	69	1	0
	Cplx+Psep	137	1	0	1	69	1	0
	All	137	1	0	1	69	1	0
30	Cplx	262	0	0	2	344	22	0
	Cplx+Umic	372	0	0	2	344	22	0
	Cplx+Psep	375	0	0	2	344	1	0
	All	373	0	0	2	344	1	0

Table B.9: Number of each type of cuts added by CPLEX for each variant for instances 21 to 30

inst	variant	Cover	GCover	FCover	Frac	Mir	ZeroHalf	LiftProj
31	Cplx	182	0	0	4	216	3	0
	Cplx+Umic	86	0	0	1	216	3	0
	Cplx+Psep	141	0	0	1	216	1	0
	All	141	0	0	1	216	1	0
32	Cplx	140	0	0	3	205	2	0
	Cplx+Umic	217	0	0	2	1	2	0
	Cplx+Psep	139	0	0	2	2	2	0
	All	121	0	0	2	1	2	0
33	Cplx	386	0	0	5	13	1	0
	Cplx+Umic	73	0	0	1	1	1	0
	Cplx+Psep	139	0	0	1	1	3	0
	All	66	0	0	1	1	3	0
34	Cplx	153	3	0	2	48	4	0
	Cplx+Umic	148	2	0	1	4	1	0
	Cplx+Psep	72	3	0	1	5	2	0
	All	49	1	0	1	5	1	0
35	Cplx	272	0	0	4	27	2	0
	Cplx+Umic	175	0	0	2	2	2	0
	Cplx+Psep	177	0	0	2	2	2	0
	All	176	0	0	2	2	2	0
36	Cplx	159	0	0	4	438	4	0
	Cplx+Umic	320	0	0	2	438	4	0
	Cplx+Psep	260	0	0	2	438	1	0
	All	260	0	0	2	438	1	0
37	Cplx	248	0	0	5	154	8	0
	Cplx+Umic	357	0	0	2	154	8	0
	Cplx+Psep	1	0	0	2	1	2	0
	All	1	1	2	1	2	2	0
38	Cplx	27	0	0	0	0	0	0
	Cplx+Umic	31	0	0	0	0	1	0
	Cplx+Psep	37	0	0	0	0	1	0
	All	31	0	0	0	3	2	0
39	Cplx	190	2	0	3	14	1	0
	Cplx+Umic	355	3	0	1	14	1	1
	Cplx+Psep	1	3	0	1	14	1	1
	All	1	3	0	1	14	1	1
40	Cplx	2	1	0	3	0	0	1
	Cplx+Umic	2	1	0	3	0	0	1
	Cplx+Psep	2	1	0	3	0	0	1
	All	2	1	0	3	0	0	1

Table B.10: Number of each type of cuts added by CPLEX for each variant for instances 31 to 40

inst	variant	Cover	GCover	FCover	Frac	Mir	ZeroHalf	LiftProj
41	Cplx	565	0	0	5	45	0	0
	Cplx+Umic	371	0	0	2	45	2	0
	Cplx+Psep	378	0	0	2	45	1	0
	All	373	0	0	2	45	2	0
42	Cplx	867	0	0	3	27	8	0
	Cplx+Umic	450	0	0	3	27	8	0
	Cplx+Psep	537	0	1	3	1	8	1
	All	520	0	1	2	1	8	1
43	Cplx	566	0	0	3	250	0	1
	Cplx+Umic	538	0	0	1	1	0	1
	Cplx+Psep	540	0	0	1	1	0	1
	All	538	0	0	1	1	0	1
44	Cplx	289	0	0	6	9	0	1
	Cplx+Umic	188	0	0	2	9	0	1
	Cplx+Psep	113	0	0	3	9	0	1
	All	109	0	0	3	9	0	1
45	Cplx	900	0	0	3	1	1	0
	Cplx+Umic	539	0	0	1	1	1	0
	Cplx+Psep	541	1	0	1	1	1	0
	All	540	1	0	1	1	1	0
46	Cplx	76	0	0	1	39	2	0
	Cplx+Umic	231	0	0	1	2	2	0
	Cplx+Psep	140	0	0	1	1	1	0
	All	140	0	0	1	1	1	0
47	Cplx	108	0	0	5	206	1	0
	Cplx+Umic	236	0	0	2	206	1	0
	Cplx+Psep	141	0	0	2	206	1	0
	All	141	0	0	2	206	1	0
48	Cplx	12	0	0	4	388	1	0
	Cplx+Umic	234	0	0	2	1	1	0
	Cplx+Psep	139	0	0	2	1	1	0
	All	139	0	0	2	1	1	0
49	Cplx	351	0	0	7	33	9	0
	Cplx+Umic	216	2	0	1	1	1	0
	Cplx+Psep	134	2	0	3	3	3	0
	All	134	2	0	3	3	3	0
50	Cplx	360	0	0	4	39	2	0
	Cplx+Umic	236	0	0	2	39	2	0
	Cplx+Psep	142	0	0	1	39	2	0
	All	142	0	0	1	39	2	0

Table B.11: Number of each type of cuts added by CPLEX for each variant for instances 41 to 50

inst	variant	Cover	GCover	FCover	Frac	Mir	ZeroHalf	LiftProj
51	Cplx	104	0	0	2	30	1	0
	Cplx+Umic	352	0	0	2	30	1	0
	Cplx+Psep	61	0	0	1	30	1	0
	All	61	0	0	1	30	1	0
52	Cplx	73	0	0	5	147	10	0
	Cplx+Umic	130	0	0	2	2	1	0
	Cplx+Psep	38	0	0	1	1	1	0
	All	27	0	0	1	1	1	0
53	Cplx	203	0	1	3	220	27	1
	Cplx+Umic	354	0	1	2	220	2	1
	Cplx+Psep	258	2	1	2	220	1	1
	All	258	2	1	2	220	1	1
54	Cplx	313	0	0	2	288	1	1
	Cplx+Umic	350	0	0	1	288	1	1
	Cplx+Psep	262	0	0	1	288	1	1
	All	262	0	0	1	288	1	1
55	Cplx	117	0	0	4	430	54	0
	Cplx+Umic	351	0	0	1	430	54	0
	Cplx+Psep	262	0	0	1	430	54	0
	All	262	0	0	1	430	54	0
56	Cplx	587	0	0	3	40	1	0
	Cplx+Umic	587	0	0	1	40	1	0
	Cplx+Psep	63	0	0	1	40	1	0
	All	63	0	0	1	40	1	0
57	Cplx	20	0	0	3	0	0	0
	Cplx+Umic	449	0	0	2	0	0	0
	Cplx+Psep	476	1	0	2	2	0	0
	All	459	1	0	2	2	0	0
58	Cplx	805	0	0	4	46	0	0
	Cplx+Umic	512	0	0	1	46	0	0
	Cplx+Psep	541	0	0	1	46	0	1
	All	513	0	0	1	46	0	1
59	Cplx	417	0	0	5	41	0	0
	Cplx+Umic	471	0	0	3	41	0	0
	Cplx+Psep	381	0	0	2	41	0	0
	All	381	0	0	2	41	0	0
60	Cplx	326	0	0	3	235	1	0
	Cplx+Umic	371	0	0	1	1	1	1
	Cplx+Psep	321	0	0	1	1	1	1
	All	318	0	0	1	1	1	1

Table B.12: Number of each type of cuts added by CPLEX for each variant for instances 51 to 60

C Appendix for the two-phase algorithm

Table of contents

C.1 Result tables

	CPLEX				RCSPP		HA*		BORWin		
instance	value	gap	time	#nodes	value	time	value	time	value	#iter	time
1	−27092.4	opt	14.0	24213	-	3969.29	-	3600.33	−27092.4	2950	1.43
2	42334.7	opt	1.0	9653	42334.7	3.77	-	3600.41	42334.7	2482	1.4
3	2480.68	opt	104.0	97567	-	3842.11	-	3600.36	2480.68	5475	3.49
4	1556.68	opt	0.0	531	-	3806.62	-	3600.48	1556.68	316	0.51
5	108120.0	opt	0.0	863	-	3620.11	-	3600.34	108120.0	860	1.64
6	−2550.61	opt	0.0	630	-	3963.0	-	3600.44	−2550.61	310	0.87
7	3031.07	inf	0.0	77	-	3857.97	-	3600.3	3031.07	2753	2.64
8	13903.4	inf	0.0	2487	-	3654.9	-	3600.28	13903.4	9644	12.12
9	−81458.9	inf	22.0	9845	-	3791.48	-	3600.2	−81458.9	4507	5.87
10	−622046.0	0.17	3599.0	164270	-	3618.49	-	3600.2	−622046.0	784	3.22
11	41736.5	opt	0.0	0	41736.5	112.82	-	3600.52	41736.5	17	1.56
12	−32130.1	opt	87.0	10279	-	3798.79	-	3600.14	−32130.1	582	3.86
13	−185784.0	1.58	3599.0	184953	-	3699.51	-	3600.19	−185784.0	264	2.43
14	19048.7	opt	3.0	2537	19048.7	29.5	-	3600.39	19048.7	2421	0.77
15	4876.22	opt	0.0	0	4876.22	0.01	4876.22	0.07	4876.22	7	0.08
16	12856.2	opt	0.0	1155	12856.2	0.06	12856.2	1738.4	12856.2	1832	1.17
17	14970.0	opt	0.0	435	14970.0	0.34	14970.0	606.87	14970.0	1143	0.37
18	12647.3	opt	0.0	1801	12647.3	0.07	-	3600.33	12647.3	1845	0.66
19	9276.1	opt	0.0	159	9276.1	0.04	9276.1	157.2	9276.1	815	0.31
20	12600.0	opt	0.0	45	12600.0	0.31	12600.0	69.8	12600.0	601	0.34
21	0.0	opt	0.0	0	0.0	0.0	0.0	0.01	0.0	1	0.01
22	21996.5	opt	0.0	0	21996.5	0.04	21996.5	0.01	21996.5	0	0.0
23	96870.5	opt	14.0	32704	96870.5	9.82	96870.5	3167.85	96870.5	2897	3.02
24	18974.6	opt	0.0	0	18974.6	0.02	18974.6	0.01	18974.6	0	0.0
25	197441.0	opt	0.0	0	197441.0	2.0	197441.0	477.82	197441.0	2058	2.12
26	21588.0	opt	0.0	0	21588.0	0.04	21588.0	0.01	21588.0	5	0.36
27	95900.6	opt	0.0	0	95900.6	0.02	95900.6	0.01	95900.6	0	0.01
28	102450.0	opt	0.0	0	102450.0	0.02	102450.0	0.01	102450.0	0	0.01
29	72778.7	inf	4.0	11675	-	3690.14	-	3600.52	72778.7	31422	28.62
30	−5397.97	opt	0.0	0	−5397.97	0.01	−5397.97	0.01	−5397.97	0	0.0
31	705.98	opt	0.0	0	705.98	0.01	705.98	0.01	705.98	6	0.06
32	24792.1	opt	0.0	0	24792.1	0.01	24792.1	0.01	24792.1	0	0.0
33	3079.3	opt	0.0	0	3079.3	0.05	3079.3	0.01	3079.3	0	0.0
34	62604.5	opt	0.0	0	62604.5	355.76	62604.5	1147.09	62604.5	344	0.64
35	100832.0	opt	0.0	0	100832.0	0.02	100832.0	0.01	100832.0	0	0.01
36	28104.4	opt	0.0	0	28104.4	0.49	28104.4	0.21	28104.4	17	0.21
37	421369.0	opt	0.0	177	-	3723.65	-	3600.41	421369.0	95	0.46
38	−4281.16	opt	0.0	0	−4281.16	95.53	-	3600.49	−4281.16	1776	1.47
39	−31910.4	opt	0.0	4955	-	3658.38	-	3600.35	−31910.4	3130	3.14
40	18323.5	opt	0.0	71	18323.5	42.47	-	3600.58	18323.5	594	0.84
41	−2480.26	opt	0.0	0	−2480.26	0.0	−2480.26	0.01	−2480.26	26	0.11
42	4529.62	opt	0.0	0	4529.62	0.03	4529.62	0.01	4529.62	0	0.0

Table C.1: Performance of model $M_{op\text{-}DRM}$ solved with CPLEX, the RCSPP algorithm, HA* and BORWin on EDF inspired instances 1 to 42

	CPLEX				RCSPP		HA*		BORWin		
instance	value	gap	time	#nodes	value	time	value	time	value	#iter	time
43	269050.0	opt	0.0	0	269050.0	0.03	269050.0	0.01	269050.0	0	0.0
44	919601.0	sub	5.0	23903	-	4535.04	-	3600.5	919605.0	2198	1.19
45	287722.0	inf	0.0	26	-	3689.87	-	3600.36	287722.0	72	0.68
46	52834.5	opt	0.0	0	52834.5	0.07	52834.5	0.04	52834.5	0	0.01
47	459095.0	opt	0.0	0	459095.0	0.04	459095.0	0.01	459095.0	0	0.0
48	462869.0	inf	1.0	1571	-	3780.74	-	3600.45	462869.0	693	0.82
49	42341.8	opt	0.0	0	42341.8	0.06	42341.8	0.03	42341.8	0	0.0
50	4905.83	opt	1.0	8813	-	3608.06	-	3600.43	4905.83	2614	2.18
51	40192.6	opt	0.0	0	40192.6	0.01	40192.6	0.02	40192.6	0	0.0
52	48378.6	opt	0.0	280	-	3678.46	-	3600.39	48378.6	784	0.55
53	28142.9	opt	0.0	0	28142.9	1.75	28142.9	0.19	28142.9	9	0.03
54	3838650.0	sub	0.0	0	3838990.0	75.46	-	3600.46	3838990.0	1687	0.78
55	9577.87	opt	0.0	425	-	3606.21	-	3600.73	9577.87	900	0.8
56	28471.9	opt	0.0	0	28471.9	0.01	28471.9	0.03	28471.9	0	0.01
57	3931900.0	inf	0.0	1177	-	3716.47	-	3600.29	3931920.0	287191	2468.04
58	750878.0	opt	0.0	544	-	3637.65	-	3600.58	750878.0	15667	24.59
59	19159.0	opt	0.0	7	19159.0	51.78	19159.0	20.16	19159.0	31	0.26
60	25720.8	opt	0.0	0	25720.8	0.05	25720.8	0.01	25720.8	0	0.0
61	−19718.7	opt	0.0	713	-	3686.43	-	3600.6	−19718.7	1555	0.88
62	12003.3	opt	0.0	30	12003.3	438.36	12003.3	21.86	12003.3	109	0.49
63	935608.0	opt	0.0	0	935608.0	0.06	935608.0	0.02	935608.0	0	0.0
64	124600.0	opt	2.0	8219	-	3624.53	-	3600.48	124600.0	15577	45.85
65	379220.0	opt	0.0	0	379220.0	0.01	379220.0	0.01	379220.0	0	0.0
66	164134.0	opt	2.0	10774	164134.0	2586.77	-	3600.61	164134.0	5171	4.63
67	703224.0	opt	0.0	0	703224.0	0.02	703224.0	0.02	703224.0	0	0.0
68	94517.5	opt	0.0	51	94517.5	0.02	94517.5	0.03	94517.5	52	0.42
69	65259.0	inf	3.0	3410	-	3672.31	-	3600.29	65259.0	5138	4.73
70	−2612.74	opt	0.0	0	−2612.74	0.0	−2612.74	0.01	−2612.74	0	0.0
71	667875.0	inf	1.0	1478	-	3750.4	-	3600.43	667875.0	1458	2.83
72	−627719.0	opt	0.0	0	-	3719.42	-	3600.33	−627719.0	84	0.8
73	48011.4	inf	0.0	24	-	3603.42	-	3600.45	48011.4	352	0.97
74	71518.2	opt	0.0	371	-	3610.12	-	3600.52	71518.2	678	1.24
75	16119.0	opt	0.0	0	16119.0	4.08	16119.0	1.42	16119.0	239	0.17
76	4976.64	opt	0.0	0	4976.64	0.07	4976.64	0.02	4976.64	0	0.01
77	−3343.57	opt	0.0	186	−3343.57	779.71	−3343.57	1525.54	−3343.57	222	0.17
78	−3209.93	opt	0.0	0	−3209.93	0.0	−3209.93	0.01	−3209.93	0	0.0
79	64146.1	opt	0.0	0	-	3742.06	-	3600.29	64146.1	3982	2.82
80	15155.4	inf	0.0	0	15155.4	147.59	15155.4	58.44	15155.4	287	0.26
81	84971.2	opt	0.0	1000	-	3711.35	-	3600.43	84971.2	252119	3148.39
82	33343.4	opt	0.0	0	-	3661.97	-	3600.54	33343.4	227	0.31
83	775.58	opt	0.0	25	775.58	0.01	775.58	0.18	775.58	142	0.18

Table C.2: Performance of model $M_{op\text{-}DRM}$ solved with CPLEX, the RCSPP algorithm, HA* and BORWin on EDF inspired instances 43 to 83

	CPLEX				RCSPP		HA*		BORWin		
instance	value	gap	time	#nodes	value	time	value	time	value	#iter	time
1	−45097.8	0.24	3599.0	5159794	-	3728.16	-	3600.2	−45097.8	230460	2318.99
2	30816.8	opt	462.0	535149	30816.8	2.66	-	3600.33	30816.8	4385	2.73
3	−12364.3	0.87	3599.0	6511928	-	3657.08	-	3600.29	−12364.3	185759	1325.15
4	−4771.67	opt	88.0	51047	-	3635.03	-	3600.44	−4771.67	13212	73.31
5	100037.0	opt	0.0	23	-	3627.04	-	3600.55	100037.0	6	0.79
6	−8330.37	opt	182.0	95280	-	3625.38	-	3600.56	−8330.37	21368	58.23
7	−25268.1	opt	0.0	494	-	3641.08	-	3600.57	−25268.1	12275	25.31
8	−15274.7	inf	0.0	105	-	3786.47	−15274.7	198.96	−15274.7	1943	3.93
9	−134802.0	0.01	3599.0	797938	-	3676.52	-	3600.33	−134885.0	5155	3601.74
10	−495393.0	opt	907.0	119597	-	3628.56	-	3600.21	−495393.0	3428	7.31
11	22498.4	opt	1.0	1079	22498.4	50.23	-	3600.42	22498.4	185	2.41
12	−15704.4	opt	91.0	19017	-	3744.61	-	3600.31	−15704.4	26	5.27
13	−362953.0	0.01	3599.0	749964	-	4030.62	-	3600.26	−362953.0	106576	3601.59
14	20036.7	opt	812.0	680819	20036.7	28.34	-	3600.42	20036.7	4734	1.65
15	3060.83	opt	0.0	38	3060.83	0.02	3060.83	1648.99	3060.83	1214	0.47
16	9849.72	opt	12.0	34012	9849.72	0.07	-	3600.39	9849.72	2722	1.3
17	8633.68	opt	2.0	9440	8633.68	0.17	-	3600.51	8633.68	10660	6.11
18	10167.2	0.4	3599.0	1124478	10167.2	0.07	-	3600.51	10167.2	7908	3.63
19	8617.41	opt	1.0	5022	8617.41	0.04	8617.41	3360.88	8617.41	1965	0.91
20	8781.47	opt	13.0	34045	8781.47	0.64	-	3600.52	8781.47	4018	1.74
21	0.0	opt	0.0	0	0.0	0.0	0.0	0.01	0.0	1	0.01
22	21996.5	opt	0.0	0	21996.5	0.05	21996.5	0.01	21996.5	0	0.0
23	95051.4	0.63	3599.0	833133	95051.4	30.72	-	3600.4	95051.4	10440	31.82
24	18974.6	opt	0.0	0	18974.6	0.02	18974.6	0.01	18974.6	0	0.0
25	196839.0	opt	0.0	0	196839.0	1.83	196839.0	758.65	196839.0	1879	1.94
26	21459.5	opt	0.0	0	21459.5	0.04	21459.5	0.01	21459.5	2	0.19
27	95900.6	opt	0.0	0	95900.6	0.03	95900.6	0.01	95900.6	0	0.0
28	102450.0	opt	0.0	0	102450.0	0.03	102450.0	0.01	102450.0	0	0.0
29	62292.3	inf	2.0	6388	-	3626.06	-	3600.41	62292.3	55078	93.38
30	−5397.97	opt	0.0	0	−5397.97	0.01	−5397.97	0.01	−5397.97	0	0.0
31	99.34	opt	0.0	0	99.34	0.01	99.34	0.02	99.34	97	0.3
32	24792.1	opt	0.0	0	24792.1	0.01	24792.1	0.01	24792.1	0	0.0
33	3079.3	opt	0.0	0	3079.3	0.06	3079.3	0.01	3079.3	0	0.0
34	59431.0	opt	0.0	0	59431.0	153.67	-	3600.5	59431.0	3689	2.18
35	100832.0	opt	0.0	0	100832.0	0.03	100832.0	0.01	100832.0	0	0.0
36	27052.8	opt	0.0	0	27052.8	0.44	-	3600.66	27052.8	427	0.48
37	406009.0	sub	1.0	670	-	3662.9	-	3600.22	406023.0	134167	3601.37
38	−4316.46	opt	0.0	0	−4316.46	85.35	-	3600.57	−4316.46	5271	3.26
39	−42097.1	opt	4.0	17021	-	3695.39	-	3600.55	−42097.1	3445	2.55
40	15916.1	opt	0.0	124	15916.1	79.01	-	3600.5	15916.1	2444	2.87
41	−3187.16	opt	0.0	0	−3187.16	0.0	−3187.16	0.01	−3187.16	97	0.15
42	4529.62	opt	0.0	0	4529.62	0.04	4529.62	0.01	4529.62	0	0.0

Table C.3: Performance of model $M_{op\text{-}DRM}$ solved with CPLEX, the RCSPP algorithm, HA* and BORWin on EDF inspired instances 1 to 42 with low price variation

	CPLEX				RCSPP		HA*		BORWin		
instance	value	gap	time	#nodes	value	time	value	time	value	#iter	time
43	269050.0	opt	0.0	0	269050.0	0.04	269050.0	0.01	269050.0	0	0.0
44	917201.0	0.06	3599.0	8558238	-	3866.17	-	3600.4	917035.0	190950	3601.38
45	282424.0	sub	3.0	5610	-	3627.51	-	3600.49	282425.0	3858	2.95
46	52834.5	opt	0.0	0	52834.5	0.06	52834.5	0.03	52834.5	0	0.01
47	459095.0	opt	0.0	0	459095.0	0.05	459095.0	0.01	459095.0	0	0.0
48	452277.0	0.04	3599.0	5319419	-	3625.98	-	3600.47	452318.0	123282	2254.47
49	42341.8	opt	0.0	0	42341.8	0.06	42341.8	0.02	42341.8	0	0.0
50	−5494.62	4.65	3599.0	1298359	-	3671.95	-	3600.47	−5494.62	91615	527.29
51	40192.6	opt	0.0	0	40192.6	0.01	40192.6	0.02	40192.6	0	0.0
52	42283.9	opt	10.0	34235	-	3641.65	-	3600.54	42283.9	6194	4.61
53	27442.4	opt	0.0	0	27442.4	0.66	27442.4	0.1	27442.4	31	0.04
54	3798460.0	sub	0.0	0	3798740.0	20.38	-	3600.47	3798740.0	31065	22.19
55	−1957.72	opt	2322.0	898717	-	3626.03	-	3600.54	−1957.72	7688	19.39
56	28471.9	opt	0.0	0	28471.9	0.01	28471.9	0.02	28471.9	0	0.01
57	3922410.0	sub	0.0	352	-	3831.41	-	3600.29	3922560.0	76710	395.57
58	716594.0	opt	2930.0	844090	-	3692.37	-	3600.62	714666.0	193649	3601.47
59	17577.3	opt	2.0	1768	17577.3	5.06	17577.3	52.03	17577.3	514	1.34
60	25720.8	opt	0.0	0	25720.8	0.04	25720.8	0.01	25720.8	0	0.0
61	−34801.1	inf	9.0	40790	-	3680.74	-	3600.56	−34801.1	14291	23.78
62	10607.8	opt	16.0	13431	10607.8	290.81	10607.8	2028.97	10607.8	3113	8.36
63	935608.0	opt	0.0	0	935608.0	0.05	935608.0	0.02	935608.0	0	0.0
64	95093.1	inf	0.0	0	-	3632.43	-	3600.61	95093.1	286	0.53
65	379220.0	opt	0.0	0	379220.0	0.01	379220.0	0.01	379220.0	0	0.0
66	138473.0	0.37	3599.0	651399	138473.0	325.18	-	3600.7	138473.0	48951	381.27
67	703224.0	opt	0.0	0	703224.0	0.02	703224.0	0.02	703224.0	0	0.0
68	94516.4	opt	0.0	57	94516.4	0.02	94516.4	0.02	94516.4	32	0.62
69	66301.3	inf	7.0	4272	-	3659.77	-	3600.33	66301.3	2960	3.43
70	−2612.74	opt	0.0	0	−2612.74	0.0	−2612.74	0.01	−2612.74	0	0.0
71	618927.0	opt	16.0	16554	-	3665.96	-	3600.52	618927.0	24436	111.57
72	−842932.0	sub	2.0	588	-	3608.11	-	3600.32	−842891.0	2503	4.41
73	30603.6	opt	27.0	39539	-	3652.74	-	3600.37	30603.6	23572	48.14
74	68651.0	0.1	3599.0	5690332	-	3663.35	-	3600.73	68651.0	8209	15.63
75	15102.9	opt	0.0	0	15102.9	4.68	-	3600.54	15102.9	4086	3.91
76	4976.64	opt	0.0	0	4976.64	0.06	4976.64	0.02	4976.64	0	0.0
77	−6975.34	opt	1.0	1676	−6975.34	57.44	-	3600.63	−6975.34	1202	1.35
78	−3209.93	opt	0.0	0	−3209.93	0.0	−3209.93	0.01	−3209.93	0	0.0
79	57148.6	opt	524.0	446548	-	3671.69	-	3600.48	57148.6	176537	1347.51
80	14650.7	opt	0.0	0	14650.7	31.81	-	3600.64	14650.7	6600	5.74
81	69986.4	opt	32.0	73328	-	3705.61	-	3600.38	69986.4	147543	1148.09
82	31375.2	sub	0.0	0	-	3609.26	-	3600.6	31375.9	95735	476.4
83	164.95	opt	0.0	600	164.95	0.01	164.95	1.59	164.95	492	0.47

Table C.4: Performance of model $M_{op\text{-}DRM}$ solved with CPLEX, the RCSPP algorithm, HA* and BORWin on EDF inspired instances 43 to 83 with low price variation

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Université de Toulouse

Titre : Modèles et algorithmes pour l'optimisation de la production hydro-électrique

Mots clés : Optimisation de la production hydro-électrique, Programmation linéaire en nombres entiers, Algorithmes de graphes, Etude polyédrale, Programmation non-linéaire

Résumé : Le problème de gestion de production hydro-électrique (HUC) est un problème difficile, qui joue un rôle important dans la gestion de production électrique journalière à EDF. Dans cette thèse, nous étudions différents modèles et algorithmes pour résoudre un cas particulier du problème HUC à une usine (1-HUC). Etudier ce cas particulier présente plusieurs intérêts. D'une part, il existe des instances réelles à une usine mal résolues par les approches actuelles. D'autre part, cela nous permet d'étudier plus spécifiquement deux sources de difficultés séparément. L'une provient de la présence de non-linéarités, notamment la puissance qui est une fonction non-convexe et non-concave du débit d'eau et du volume des réservoirs. L'autre est due à l'ensemble des contraintes hydrauliques, notamment des volumes minimaux et maximaux ainsi que des volumes cibles des réservoirs.
Dans une première partie, différentes alternatives de modélisation des non-linéarités du 1-HUC, plus particulièrement sur la puissance, sont proposées. L'objectif est d'identifier un modèle pouvant être résolu efficacement avec un bon compromis entre temps de calcul et précision. Les sept alternatives proposées couvrent un large spectre de familles de modélisation. Elles sont comparées sur un jeu d'instances présentant des variations sur cinq caractéristiques qui impactent la résolution. Cette étude comparative permet d'identifier trois types de modèles pertinents: un modèle avec des fonctions polynomiales de degré 2, un modèle avec une fonction linéaire par morceaux, et un modèle utilisant un ensemble fini de débits. Ce dernier modèle étant similaire à la modélisation actuelle à EDF, nous proposons dans la suite des algorithmes dédiés à celui-ci.
Dans une deuxième partie, une étude polyédrale est proposée pour améliorer la résolution du problème 1-HUC. L'idée est de focaliser sur le coeur combinatoire, ce qui revient à considérer le lien entre les contraintes sur les volumes et l'ensemble discret des débits. Pour celà, nous définissons une variante du problème du sac-à-dos avec chaines de précédence et poids symétriques (SCPKP). Pour le SCPKP, nous définissons des conditions nécessaires de facettes, qui sont aussi prouvées suffisantes dans certains cas. Un algorithme de branch-and-cut en deux phases s'appuyant sur ces conditions et sur l'aspect symétrique du SCPKP est mis au point. L'efficacité de cet algorithme est ensuite montrée numériquement face à des algorithmes de l'état de l'art. Les résultats de cette analyse polyédrale du SCPKP, ainsi que l'algorithme de résolution proposé sont ensuite étendus au problème 1-HUC.
Dans une troisième partie, une technique de résolution efficace est proposée pour prendre en compte les contraintes hydrauliques en s'appuyant sur la représentation du problème 1-HUC par un graphe. Cela permet de se ramener à un cas particulier du problème de plus court chemin avec fenêtres de ressource (RWSPP). Nous proposons deux algorithmes de graphes. Le premier algorithme est une variante exacte de l'algorithme A*, utilisant une borne duale dédiée au problème 1-HUC. En comparaison avec deux approches de l'état de l'art, nous montrons numériquement que cet algorithme est plus efficace pour traîter un cas spécifique du 1-HUC. L'objectif du second algorithme est de prendre en compte davantage de contraintes hydrauliques. Le principe s'appuie sur le concept d'optimisation bi-objectif pour lequel le second objectif correspond à une relaxation du volume d'eau. L'avantage par rapport à une optimisation bi-objectif classique est qu'il est possible d'utiliser les volumes minimaux et maximaux pour réduire l'espace de recherche et diriger l'énumération de solutions. Nous montrons numériquement, sur un grand jeu d'instances réelles, que cet algorithme est plus performant que trois approches de l'état de l'art. Même si cet algorithme a été conçu pour résoudre le 1-HUC, nous le définissons de manière générique pour tout RWSPP avec une resource.

Title: Models and algorithms for the Hydro Unit Commitment problem

Key words: Hydro Unit Commitment, Mixed-integer linear programming, Graph algorithms, Polyhedral analysis, Non-linear programming

Abstract: The Hydro Unit Commitment problem (HUC) is a difficult problem playing a major role in the scheduling of daily electricity production at EDF. In this thesis, we study different models and algorithms to solve the special case of the single-unit HUC problem (1-HUC). Studying this case is relevant for the following reasons. On the one hand, there are real world instances of the 1-HUC problem which cannot be solved efficiently by current approaches. On the other hand, it makes it possible to study individually two particular sources of difficulty. One stems from the presence of non-linearities, in particular the power which is a non-convex non-concave function of the flow and the reservoirs' volume. The other is due to the set of hydraulic constraints, specifically the volume minimum and maximum bounds, as well as target volumes for the reservoirs.
In a first part, modeling alternatives for the non-linear 1-HUC, focusing on the power function, are proposed. The aim is to identify a model which can be solved efficiently, with a good trade-off between computational time and precision. The seven proposed modeling alternatives cover a large panel of modeling families. These models are compared on a set of instances with variations on five features that impact the solution. This comparative study enables us to identify three efficient types of model: a model with polynomial functions of degree 2, a model with a piecewise linear function, and a model using a finite set of flows. As the latter model is similar to the current model at EDF, in the following we present algorithms dedicated to it.
In the second part, a polyhedral study is proposed to improve the solving approach of the 1-HUC problem. The idea is to focus on the combinatorial aspects, which means considering the relationship between the bounds on volumes and the discrete set of flows. For this purpose, we introduce a variant of the knapsack problem, with Symmetric weight and Chain Precedences (SCPKP). For the SCPKP, we characterize necessary facet-defining conditions, which are also proven to be sufficient in some cases. A two-phase branch-and-cut algorithm based on these conditions and on the symmetric aspect of the SCPKP is devised. The efficiency of this algorithm is then shown experimentally against state-of-the-art algorithms. The results of this polyhedral study of the SCPKP, as well as the proposed algorithms, are then extended to the 1-HUC problem.
In the third part, an efficient solving technique based on a graph representation of the 1-HUC problem is proposed, taking into account of the hydraulic constraints. It appears that the 1-HUC problem is a special case of the Shortest Path Problem with Resource Windows (RWSPP). We propose two graph algorithms. The first one is an exact variant of the A* algorithm, using a dual bound dedicated to the 1-HUC problem. In comparison with two state-of-the-art approaches, we show numerically that this algorithm is more efficient for handling a specific case of 1-HUC. The aim of the second algorithm is to take into account a wider set of hydraulic constraints. The idea is based on the concept of bi-objective optimization, for which the second objective corresponds to a relaxation of the volume. The advantage compared to a classical bi-objective optimization is that it is possible to use the minimum and maximum bounds on the volume to reduce the search space and to guide the enumeration of solutions. We show numerically, on a large set of real instances, that this algorithm outperforms three state-of-the-art approaches. Although this algorithm was designed to solve the 1-HUC problem, it is defined in a generic way for any RWSPP with a single resource.