Handling polyhedral symmetries with a dedicated Branch&Cut: application to a knapsack variant

Alexandre Heintzmann, Pascale Bendotti, Cécile Rottner

Handling polyhedral symmetries with a dedicated Branch&Cut: application to a knapsack variant

Alexandre Heintzmann, Pascale Bendotti, and Cécile Rottner

EDF Lab Saclay, 7 bd Gaspard Monge, 91120 Palaiseau, France.
{alexandre.heintzmann, pascale.bendotti, cecile.rottner} @edf.fr

Abstract. In this paper, we define a new variant of the knapsack problem, the Symmetric-weight Chain Precedence Knapsack problem (SCPKP). The (SCPKP) is the core structure of the Hydro Unit Commitment problem, the latter being a production scheduling problem relative to hydroelectric plants. The (SCPKP) is shown to be NP-hard. Polyhedral symmetries, featured by the (SCPKP), are introduced as a generalization of the classical symmetries applying only on the constraints without restricting the values of two symmetric solutions to be equal. A polyhedral study focuses on inequalities with 0-1 coefficients. Necessary facet-defining conditions are described through a new structure, called pattern, encoding the polyhedral symmetries of the (SCPKP). A dedicated two-phase Branch & Cut scheme is defined to exploit the symmetries on the pattern inequalities. Experimental results demonstrate the efficiency of the proposed scheme in particular with respect to default CPLEX and a family of cover inequalities related to the Precedence Knapsack Problem.

Keywords: Polyhedral study · Precedence constraints · Cover inequalities.

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 $w_{ij} \geq 0$ and value v_{ij}. Within each group, order constraints are such that any item (i, j) can be selected provided item $(i, j-1)$ is selected, thus inducing chain precedence constraints. Let C be the maximum capacity. The Chain Precedence Knapsack problem (CPKP) is to maximize the total value of the selected items, while the chain precedence constraints are verified, and the total weight of the selected items is less than or equal to C. The Symmetric-weight Chain Precedence Knapsack problem (SCPKP) is a (CPKP) where item (i, j) has 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.

The motivation for studying the (SCPKP) is that it is the core structure of the Hydro Unit Commitment (HUC) problem [8], which is a production scheduling problem relative to hydroelectric plants.

The Knapsack Problem (KP) and its variants have been widely studied in the literature [11]. The (SCPKP) and the (CPKP) have not been studied yet,

but can be related to some classical variants of the knapsack problem. As the chain precedence constraints are a special case of precedence constraints, the (CPKP) is a direct special case of the Precedence Knapsack Problem (PKP) [5]. Similarly as disjunctive constraints can be used alternatively for the (CPKP), the (CPKP) is also a special case of the Disjunctive Knapsack Problem (DKP) [20].

In this paper, the (SCPKP) is shown to be NP-hard. A compact formulation is defined with its corresponding polytope. A literature review of facet-defining inequalities for problems related to the (SCPKP) is exposed. The main contributions are the polyhedral study of the (SCPKP), extending the preliminary work in [9], and a two-phase Branch & Cut (B&C) scheme. A new structure, called pattern, is defined to embed the symmetries of the (SCPKP). New inequalities, associated with the patterns and covering all the facets of the (SCPKP) with 0-1 coefficients are introduced. Necessary facet-conditions are defined for these inequalities. An algorithm to generate patterns verifying these conditions is described and is used as pre-processing in the first phase of the B&C scheme. The separation algorithm, involved in the second phase of the B&C scheme, reduces to solve a maximum matching problem for the pool of the first phase generated patterns. To evaluate the efficiency of the corresponding inequalities, a numerical comparison shows that pattern inequalities largely outperform CPLEX generated cuts, and also improve standard lifted minimal (induced) cover inequalities for variants of the KP.

In **Section 1** the complexity of the (SCPKP) is stated and some polyhedral results from the literature for variants of the KP are reported. Polyhedral symmetries are introduced. In **Section 2** patterns are defined as well as inequalities and necessary facet-defining conditions are provided. In **Section 3** the two-phase B&C scheme is described. In **Section 4** experimental results are presented. Proofs and algorithmic details can be found in an online companion paper [10].

1 The Symmetric-weight Chain Precedence Knapsack

In this section, we first state the NP-hardness of the (SCPKP). A literature review of related problems is done, and first polyhedral properties are provided and polyhedral symmetries are introduced.

We first identify two special cases where the (SCPKP) is easy to solve.

Property 1 *If $I = 1$ or $J = 1$, the (SCPKP) can be solved in polynomial time.*

In the following, we then only consider instances of the (SCPKP) with $I \geq 2$ and $J \geq 2$. The NP-hardness of the (SCPKP) can be proven by reduction from the Subset Sum Problem.

Theorem 1 *The (SCPKP) is NP-hard when $I \geq J$*

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) F_{SCPK} as follows.

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

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

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

In this formulation, the objective function is to maximize the total value of the selected items. Inequality (1) is the capacity constraint, inequalities (2) correspond to the chain precedence constraints. We define the polytope $P_{(SCPKP)}$ as the convex hull of the feasible solutions of the (SCPKP):

$$P_{(SCPKP)} = \text{conv}\Big\{x \in \{0,1\}^n : x \text{ satisfies } (1) - (2)\Big\}.$$

The proposed formulation is the so-called *incremental formulation* [16]. It is also possible to define a *multiple choice* formulation, featuring disjunctive constraints instead of chain precedence constraints (2). Both formulations yield the same LP relaxation [6], but we do not further detail the multiple choice formulation, as it is not necessary for the purpose of this paper.

We can remark that the (SCPKP) features symmetries. Indeed, items have symmetric weights, the chain precedence constraints (2) are the same for each group, and the capacity constraint (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 1 *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, thus generalizing the classical concept of symmetry, which also restricts the objective values of two symmetric solutions to be equal. Therefore classical methods to handle symmetries [15] may not apply to polyhedral symmetries.

For example, polyhedral symmetries featured by the SCPK correspond to the action of the symmetric group acting on the columns of the solution matrix, *i.e.*, any permutation of the columns of the solution matrix is a polyhedral symmetry. The most recent developments to handle such symmetries involve lexicographical ordering [11, 3, 4], symmetric branching disjunctions [17] or variable aggregation [12]. Such techniques rely on the symmetrical aspects of the objective function and therefore do not apply to polyhedral symmetries of the SCPK.

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.

1.1 Related knapsack polytopes

The (SCPKP) is a knapsack variant. In this section, we present valid inequalities of related knapsack problems. We focus on generalizations of the (SCPKP), namely the (PKP) and the (DKP).

Disjunctive constrained Knapsack Problem Five families of inequalities containing facet-defining inequalities have been reported for the (DKP) [20]: the clique inequalities; the cover inequalities; odd-cycle and hypergraph inequalities, the clique-cover inequalities; the clique-cover-partition inequalities. Extensions of cover inequalities such as clique-cover inequalities and clique-cover-partition inequalities 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, therefore the aforementioned inequalities cannot be adapted to the (SCPKP).

Precedence constrained knapsack polytope The (PKP) [5] is defined as a binary knapsack problem with additional prcedence constraints. Classical inequalities from the binary knapsack polytope, namely minimal cover and related inequalities [1], have been extended to the (PKP) [5] as *Minimal Induced Cover* (MIC) inequalities [5, 18, 14]. These MIC inequalities have been enhanced with a lifting procedure [18], to obtain Downlifted and Uplifted MIC inequalities. Such inequalities can be adapted to the (SCPKP), by introducing coefficients 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 2 *Let U be a MIC for the (SCPKP), for all $(i,j) \in U_p$, the downlifting procedure yields $\alpha_{ij} = 0$.*

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.

Valid Uplifted MIC (UMIC) inequalities can be obtained with the β coefficients produced by the procedure of [7]. Such inequalities will be experimentally compared to the inequalities introduced in this paper in **Section 4**.

1.2 First polyhedral properties and definitions

Definition 2 (Full-dimensional condition (fd)) *An (SCPKP) verifies (fd) if any item (i,j), can be selected in at least one feasible solution.*

Theorem 2 *$P_{(SCPKP)}$ is full dimensional if condition (fd) holds.*

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.*

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

We define an *item set* $\mathcal{X}$ a set of items (i, j) of the (SCPKP). To account for the chain precedence constraints, we introduce the *set weights* associated with an item set $\mathcal{X}$. In the following, we only consider item sets $\mathcal{X}$ such that item $(i, j) \in \mathcal{X}$ corresponds to variable $x_{ij} \in \mathcal{V}$. For a given item set $\mathcal{X}$, and for all $i \leq I$, $j \leq J$, the set weights are

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

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

For the sake of simplicity, we introduce the k-intersection.

Definition 3 (k-intersection) *Let $\mathcal{X}$, $\mathcal{Y}$ be item sets. Item set $\mathcal{Y}$ is a k-intersection of $\mathcal{X}$ if the three following conditions hold: 1) $|\mathcal{Y} \cap \mathcal{X}| = k$, 2) for each item $(i, j) \in \mathcal{Y}$ if $(i, j') \in \mathcal{X}$ with $j' \leq j$, then $(i, j') \in \mathcal{Y}$, 3) $\sum_{(i,j)\in\mathcal{Y}} s_{ij}(\mathcal{Y}) \leq C$.*

Following this definition, a k-intersection of $\mathcal{X}$ defines a set of items that can all be selected simultaneously in a feasible solution of the (SCPKP), in which k items of $\mathcal{X}$ are selected. Consequently, if there exists a k-intersection of $\mathcal{X}$, then there is a feasible solution where k items of $\mathcal{X}$ are selected.

Note that 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 $\mathcal{X}$.

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

Clearly, the result of this property is directly due to the polyhedral symmetry, defined in **Definition 1**. Like the binary knapsack problem [1], 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 article, the polyhedral study focuses on the binary inequalities through a structure, defined as a *pattern*, to handle their symmetries.

2 Patterns 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.

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

The aim is to produce inequalities from the patterns. For this purpose, we define the item set $\mathcal{X}$ *associated with* pattern $\mathcal{P}$ and a permutation π of $\{1, \ldots, I\}$ such that $(i, j) \in \mathcal{X}$ for each $j \in S_{\pi(i)}(\mathcal{P})$. We denote $\Omega(\mathcal{P})$ the set of all item sets associated with $\mathcal{P}$. Note that $|\Omega(\mathcal{P})|$ is in general exponential.

For the remainder of **Section 2**, when referring to $\mathcal{X} \in \Omega(\mathcal{P})$, we consider without loss of generality that π is the identity permutation π_{id} if not mentioned otherwise.

We extend the definition of cardinality for a set to *cardinality* for a pattern $\mathcal{P}$ as $card(\mathcal{P}) = |\mathcal{X}|$ with $\mathcal{X} \in \Omega(\mathcal{P})$. This extension is valid by definition of an item set associated to $\mathcal{P}$.

The *rank* of a pattern $\mathcal{P}$ is the valid upper bound for the sum of variables in any item set associated with $\mathcal{P}$, as follows

$$rank(\mathcal{P}) = \max_{\mathcal{X} \in \Omega(\mathcal{P})} \Big\{ \max \sum_{(i,j) \in \mathcal{X}} x_{ij} : \text{satisfying } (1) - (2) \Big\}.$$

The rank of a pattern can be computed with a shortest path algorithm [2]. With $rank(\mathcal{P})$ and $\Omega(\mathcal{P})$, we can define pattern inequalities.

Definition 4 (Pattern inequalities) *The pattern inequalities associated with a pattern $\mathcal{P}$ are the following, for any $\mathcal{X} \in \Omega(\mathcal{P})$:*

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

By definition of the rank, pattern inequalities are valid for $P_{(SCPKP)}$. A pattern $\mathcal{P}$ is a *facet-defining pattern* if for every $\mathcal{X} \in \Omega(\mathcal{P})$, inequality $(pi(\mathcal{X}))$ is facet-defining for the (SCPKP).

2.1 Necessary facet defining conditions

In this section, we consider $\mathcal{P}$ be a pattern of rank k and an associated item set $\mathcal{X} \in \Omega(\mathcal{P})$. 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 if its sets.

Property 5 (Condition (i): no empty group) *Let $\mathcal{P}$ be a pattern. If $\mathcal{P}$ is a pattern-facet, then it verifies condition (i):*

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

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

Property 6 (Condition (ii): selection of item J) *Let $\mathcal{P}$ be a pattern and $\mathcal{X} \in \Omega(\mathcal{P})$ a variable set. If $\mathcal{P}$ is a pattern-facet, then $\mathcal{P}$ verifies condition (ii) :*

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

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

Property 7 (Condition (iii): independence from the predecessors) *Let $\mathcal{P}$ be a pattern and $\mathcal{X} \in \Omega(\mathcal{P})$ be an item set. If $\mathcal{P}$ is a pattern-facet, then $\mathcal{P}$ verifies condition (iii) :*

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

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, by solving a shortest path algorithm at most once for each variable. As these conditions are necessary, we define a *flexible pattern*, which verifies conditions (i), (ii) and (iii).

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 patterns inequalities.

Theorem 3 ($n - card(\mathcal{P})$ linearly independent points) *Let $\mathcal{P}$ be a flexible pattern and an associated pattern inequality $pi(\mathcal{X})$. Let n be the number of variables of the (SCPKP). There are at least $n - card(\mathcal{P})$ linearly independent points that verify $pi(\mathcal{X})$ to equality.*

This result will be used in the separation to generate strong pattern-inequalities.

3 Two-phase Branch&Cut algorithm for pattern inequalities

As the separation problem for pattern inequalities seems to be NP-hard, we show that a special case of this problem (namely the fixed-pattern case) can be solved in polynomial time. Based on this result, we devise a two-phase Branch&Cut algorithm.

Definition 5 (Fixed-pattern separation problem) *Consider $\widetilde{x}$ a fractional solution for the (SCPKP). Let $\mathcal{P}$ be a pattern of rank k. Finding the most violated pattern inequality associated to pattern $\mathcal{P}$ means finding the permutation π maximizing $\sum_{i=1}^{I} \sum_{j \in S_i(\mathcal{P})} \widetilde{x}_{\pi(i)j} - k$*

We show that the fixed-pattern separation problem can be done in polynomial time by solving a Maximum Matching Problem (MMP)

Property 8 *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 the (MMP) in a particular weighted bipartite graph.*

As the (MMP) is a problem that can be solved in polynomial time [13], so is the fixed-pattern separation problem. Based on this complexity result, we define a two-phase B&C scheme. The first phase generates flexible patterns as a pre-processing step. The second phase separates the inequalities associated to this predefined set of patterns within a B&C framework.

First phase: pattern generation The idea of the proposed procedure is to iteratively construct patterns so we obtain a pool P_0 of flexible patterns of non-trival rank. The generation procedure iterates until a predefined time limit is reached. At each iteration of the generation, the procedure selects randomly an integer k as the *target rank* for the pattern $\mathcal{P}$ to be constructed and first initializes a pattern $\mathcal{P}$ containing I times the set $\{J\}$. Then the goal is to iteratively modify pattern $\mathcal{P}$ until we obtain a flexible-pattern with rank k. To do so, while the cardinality of $\mathcal{P}$ is less than k, a random group i is selected and $j-1$ is added to $S_i(\mathcal{P})$, where j is the smallest index in $S_i(\mathcal{P})$. Then, once $card(\mathcal{P}) \geq k$, we update $\mathcal{P}$ in order to satisfy flexible-pattern conditions, in particular (ii) and (iii). If at the end of the iteration we obtain a flexible pattern of rank k, we add it to the pattern pool P_0.

Second phase: fixed-pattern separation at each node For each fractional point encountered in the B&C tree, we run the following separation procedure if the number of pattern inequalities added is below 100. For each pattern $\mathcal{P}$ of the pool P_0 generated in the first phase, we run the Hungarian algorithm to solve the matching problem associated to the fixed-pattern separation problem for pattern $\mathcal{P}$. For each pattern, this yields a pattern inequality. We add the most violated pattern inequality and discard the others. For each pattern, we compute an indicator corresponding to the average violation value of all added cuts associated to the pattern. In order to limit the separation time, we progressively reduce the size of pool P_0 and keep patterns with the highest average violation value.

4 Experimental results

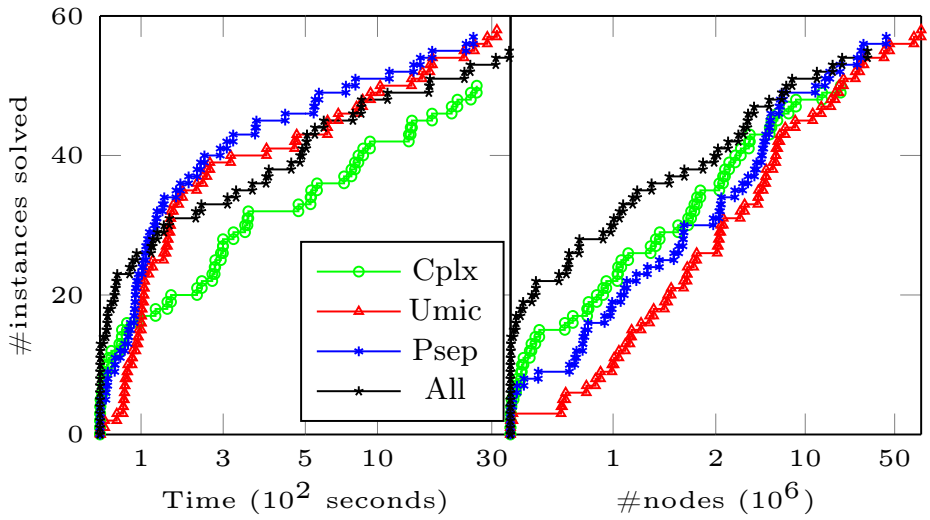
Results are computed on a single thread of a Linux machine with Intel Core i7-9850H CPU @ 2.60GHz processor, and 12 CPUs of 12 cores. Version 12.8 of CPLEX C++ API is used to solve model F_{SCPK}. We compare four sets of inequalities: Cplex's internal cuts (denoted by *Cplx*), UMIC inequalities (denoted by *Umic*), pattern inequalities (denoted by *Psep*) and the combination of all three (denoted by *All*). Each of these sets are implemented within a branch&cut (B&C) framework, limited to 3600 seconds of computational time. Pattern and UMIC inequalities are added with CPLEX's usercut callbacks.

Instances From a pool of hundred randomly 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 enabled but the cuts. As such, we obtain difficult instances with a strong enough correlation between weights and values [19]. The sixty retained instances are with $I \in \{20, 30\}$ and $J \in \{5, 10\}$. The generation of instances is further detailed in [10].

Separation Cplx corresponds to default CPLEX, and the default CPLEX separation algorithm is used. For variant Umic, we separate UMIC inequalities with the uplifting and separation algorithms described in [7]. In particular, the separation is only enabled for the root node. The separation procedure is disabled once the number of UMIC inequalities added reaches ten times the number of inequalities of F_{SCPK}. For variant Psep, the separation of the pattern inequalities is done as described in Section 3. The time limit for the first phase is fixed to 30 seconds. For Umic and Psep, the separation algorithms are implemented using UserCut Callback, and all CPLEX's cuts are disabled.

Results The results in terms of computational time (resp. number of nodes) are presented in Figure 1a (resp. 1b), representing the number of instances solved for each variant with respect to the computational time (resp. number of nodes). Note that for easy readability of Figure 1a (resp. Figure 1b), the scale is linear until 500 seconds (resp. 2 000 000 nodes), then a logarithmic scale is used. We notice on **Figure 1a** that with 100 seconds, All solves the largest number of instances and Umic the smallest number of instances. However, when the computational time reaches 100 seconds, variant Psep becomes the most efficient whereas Cplx becomes the least efficient. In Figure 1b, All is the variant developing the lowest number of nodes and Umic is the variant developing the largest number of nodes.

Interpretation Variant Cplx solves far less instances than variants Psep and Umic for large computational times, despite developing fewer nodes. This could mean that Cplx adds many more inequalities than variants Umic and Psep, which slightly reduces the number of nodes, but greatly increases the computational time. In both figures, variant All clearly dominates variant Cplx meaning that UMIC and pattern inequalities always improve the resolution. Variant All is particularly effective for instances where at least one of the variants Cplx, Umic and Psep yield cuts that drastically reduce the B&C tree. However, when no variant yield such efficient cut, similar conclusions can be drawn for All than for Cplx, i.e., too many inequalities are added, which yields larger computational times than variants Psep and Umic for instances. When comparing Umic and Psep, Figures 1a and 1b are consistent, as Psep is quicker than Umic and develops fewer nodes. These results show that our two-phase B&C scheme outperforms a standard B&C featuring state-of-the-art separation algorithms.



(a) Number of instances solved with respect to the computational time

(b) Number of instances solved with respect to the number of nodes

Fig. 1: Comparative performance of the B&C algorithms involving either Cplx, Umic, Psep or All

5 Perspectives

In this paper, two main contributions are proposed. A polyhedral study of the (SCPKP) and the two-phase B&C scheme, both revolving around the patterns introduced to handle polyhedral symmetries.

One direct extension of this work is to find efficient combinations of the considered inequalities, in order to yield an even more effective B&C algorithm. It would be relevant to study integer inequalities for the (SCPKP), as they are part of the convex hull for many Knapsack Problem variants, including the (SCPKP). The proposed patterns can also be extended to other problems, such as the use case of the Hydro Unit Commitment problem whose core structure corresponds to the (SCPKP). A promising perspective would be to generalize the two phase B&C scheme to other problems facing polyhedral symmetries.

Bibliography

[1] Balas, E.: Facets of the knapsack polytope. Mathematical programming **8**(1), 146–164 (1975)

[2] Bellman, R.: On a routing problem. Quarterly of applied mathematics **16**(1), 87–90 (1958)

[3] Bendotti, P., Fouilhoux, P., Rottner, C.: Symmetry-breaking inequalities for ilp with structured sub-symmetry. Mathematical Programming **183**, 61–103 (2020)

[4] Bendotti, P., Fouilhoux, P., Rottner, C.: Orbitopal fixing for the full (sub-) orbitope and application to the unit commitment problem. Mathematical Programming **186**, 337–372 (2021)

[5] Boyd, E.: Polyhedral results for the precedence-constrained knapsack problem. Discrete Applied Mathematics **41**(3), 185–201 (1993)

[6] Croxton, K.L., Gendron, B., Magnanti, T.L.: A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. Management Science **49**(9), 1268–1273 (2003)

[7] Espinoza, D., Goycoolea, M., Moreno, E.: The precedence constrained knapsack problem: Separating maximally violated inequalities. Discrete Applied Mathematics **194**, 65–80 (2015)

[8] Hechme-Doukopoulos, G., Brignol-Charousset, S., Malick, J., Lemaréchal, C.: The short-term electricity production management problem at EDF. Optima Newsletter **84**, 2–6 (Oct 2010)

[9] Heintzmann, A., Bendotti, P., Rottner, C.: Polyhedral study of the symmetrically weighted matrix knapsack problem. In: Proceedings of the 7th International Symposium on Combinatorial Optimization (2022)

[10] Heintzmann, A., Bendotti, P., Rottner, C.: Two-phase branch & cut for the symmetric weight matrix knapsack polytope. URL: https://hal.science/hal-03992007v1 (2023)

[11] Hojny, C., Gally, T., Habeck, O., Lüthen, H., Matter, F., Pfetsch, M.E., Schmitt, A.: Knapsack polytopes: a survey. Annals of Operations Research **292**(1), 469–517 (2020)

[12] Knueven, B., Ostrowski, J., Watson, J.P.: Exploiting identical generators in unit commitment. IEEE Transactions on Power Systems **33**(4), 4496–4507 (2017)

[13] Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)

[14] van de Leensel, R.L., Van Hoesel, C., Van de Klundert, J.: Lifting valid inequalities for the precedence constrained knapsack problem. Mathematical programming **86**(1), 161–185 (1999)

[15] Margot, F.: Symmetry in integer linear programming. 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art pp. 647–686 (2009)

[16] Markowitz, H.M., Manne, A.S.: On the solution of discrete programming problems. Econometrica: journal of the Econometric Society pp. 84–110 (1957)
[17] Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. Mathematical Programming **126**, 147–178 (2011)
[18] Park, K., Park, S.: Lifting cover inequalities for the precedence-constrained knapsack problem. Discrete Applied Mathematics **72**(3), 219–241 (1997)
[19] Pisinger, D.: Where are the hard knapsack problems? Computers & Operations Research **32**(9), 2271–2284 (2005)
[20] Salem, M.B., Taktak, R., Mahjoub, A.R., Ben-Abdallah, H.: Optimization algorithms for the disjunctively constrained knapsack problem. Soft Computing **22**(6), 2025–2043 (2018)