INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 0 (2022) 1–25
DOI: 10.1111/itor.13207

# A decomposition approach for multidimensional knapsacks with family-split penalties

Simona Mancini[a,b], Carlo Meloni[c,*] and Michele Ciavotta[d]

[a] *Department of Operations, Energy, and Environmental Management University of Klagenfurt, Klagenfurt, Austria*
[b] *Dipartimento di Ingegneria, Università di Palermo, Palermo, Italy*
[c] *Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza Università di Roma, Rome, Italy*
[d] *Dipartimento di Informatica, Sistemistica e Comunicazione Università di Milano-Bicocca, Milan, Italy*
*E-mail: simona.mancini@aau.at [Mancini]; carlo.meloni@uniroma1.it [Meloni]; michele.ciavotta@unimib.it [Ciavotta]*

Received 23 September 2021; received in revised form 18 August 2022; accepted 21 August 2022

**Abstract**

The optimization of Multidimensional Knapsacks with Family-Split Penalties has been introduced in the literature as a variant of the more classical Multidimensional Knapsack and Multi-Knapsack problems. This problem deals with a set of items partitioned in families, and when a single item is picked to maximize the utility, then all items in its family must be picked. Items from the same family can be assigned to different knapsacks, and in this situation split penalties are paid. This problem arises in real applications in various fields. This paper proposes a new exact and fast algorithm based on a specific Combinatorial Benders Cuts scheme. An extensive experimental campaign computationally shows the validity of the proposed method and its superior performance compared to both commercial solvers and state-of-the-art approaches. The paper also addresses algorithmic flexibility and scalability issues, investigates challenging cases, and analyzes the impact of problem parameters on the algorithm behavior. Moreover, it shows the applicability of the proposed approach to a wider class of realistic problems, including fixed costs related to each knapsack utilization. Finally, further possible research directions are considered.

*Keywords:* knapsack problems; discrete optimization; integer programming; decomposition methods; benders cuts

## 1. Introduction

The class of Knapsack problems belongs to the set of the most studied and challenging problems in combinatorial and integer optimization (Martello and Toth, 1990; Kellerer et al., 2004; Cacchiani et al., 2022a). The *original* Knapsack problem (KP) is still of great interest due to its applications in many contexts. Nevertheless, over the years, its basic model has been modified and extended

*Corresponding author.

providing several *variants* investigated in the literature, including the Multiple Knapsack Problem (MKP) and the Multidimensional Knapsack Problem (MdKP) (Cacchiani et al., 2022b). This paper considers a variant of both the more classical MKP and MdKP, namely the Multiple Multidimensional Knapsack Problem with Family-Split Penalties (MMdKFSP) introduced by Mancini et al. (2021) that occurs in different application areas, including resource management in distributed computing systems and service oriented architectures, logistics, and manufacturing. In this problem, the items are partitioned into *families* of interdependent entities. Such a forceful belonging is conveyed in the problem by the constraint that if one family member is chosen, all other components of the same family must be chosen as well. Furthermore, although items belonging to the same family can be allocated in different knapsacks, if this happens, family-split *penalties* are paid; thus reducing the global utility. Although most of the time it is reasonable to assume that every single item has its own individual profit, as in the basic (multiple) KP, in some cases it happens that the profit can only be defined for a family of items, rather than for each of them separately. In Chen and Zhang (2018), the classical meaning of the knapsack problem is extended, considering two examples. In the first case, several people go hiking together. In this situation, all that is necessary for putting a tend in place, such as the poles, ropes, sticks, and the tent itself has a single value that may only be obtained if every item is available. The second example deals with the delivery of large equipment or devices, which could be split into smaller parts and carried by multiple vehicles. However, no piece of equipment has an individual value, and it only makes sense to carry all the parts. Similar cases can be encountered in the manufacturing sector concerning the management of tooling of machining centers, components preparation, and kitting. The MMdKFSP also arises in resource and operations management of distributed computing applications and service oriented architectures (Mancini et al., 2021) either on a local scale (e.g., departmental servers), or on a larger scale (e.g., public cloud environments). In these contexts, items are generally used for modeling computing tasks or services, with heterogeneous resource requirements. Items are organized into families according to the software process or application they relate to, kind of requirements, or customers. In this scenario, knapsacks usually model computing nodes (e.g., virtual machines and physical servers) and are characterized by a limited capacity in terms of various resources such as CPU, RAM, storage, and networking. The items, that is, the parts of software applications waiting to be allocated and executed on the machines, have their needs with respect to the resources utilization, and when one item is chosen to maximize a global index of performance, its family must be entirely chosen as well, that is, a family cannot be partially chosen because an application might not be functional without all its components. Nevertheless, items belonging to the same family can be assigned to different knapsacks; however, in this situation, split penalties are paid to consider either the required additional organizational burdens and/or the quality-of-service reductions (Sun et al., 2016).

All these applications motivate the study of the MMdKFSP and the pursuit of suitable algorithmic solutions. In fact, notwithstanding the interest due to the real-world applications of this problem, it has not yet received adequate attention from the modeling and algorithmic point of view. To this aim, this paper intends to make a significant step forward with respect to what is proposed in the literature. More specifically, this research work is motivated by the need to improve the state-of-the-art on this subject, designing and testing new algorithms to efficiently handle large-size instances.

The main contributions of this paper can be summarized in the following: (i) it proposes a new *exact* and *rapid* solution method for the MMdKFSP based on the application of a specific Combinatorial Benders Cuts (CBC) approach that relies on an effective problem re-formulation; (ii) it introduces a specific relaxation procedure able to provide high-quality upper bounds to be used in the presented algorithmic approach that overall is fundamentally different from the standard CBC method (Benders, 1962); (iii) it reports on a campaign of computational experiments conducted on different sets of benchmark instances and a comparison to state-of-the-art algorithms from the literature and a top-level optimization commercial software, demonstrating the effectiveness of the proposed solution; (iv) it offers an experimental analysis that addresses also algorithmic flexibility and scalability issues, studies the impact of problem parameters on the algorithm behavior, and investigates challenging instances and possible extensions or generalizations. To ensure both reproducibility and repeatability of experiments, the instances generated to test the proposed algorithm are made available online for further research.

The rest of the paper is structured as follows. Section 2 offers a review of the related problems in the literature. In Section 3, the formal description of the MMdKFSP is provided, including an integer programming formulation, and useful relaxations of the problem to obtain the upper bounds. The CBC scheme, the proposed solution method, and the implementation details are illustrated in Section 4. The description and discussion of the experimental validation of the proposed method is the theme of Section 5, where an analysis of the behavior of the algorithm is reported also considering the possible extension to a wider class of problems. Finally, conclusions follow in Section 6, indicating also possible directions for future research works.

## 2. Literature review

The MMdKFSP has been introduced in Mancini et al. (2021), showing how various problems studied in the literature share some features with it. This problem, however, presents a specific structure and is involved in relevant applications that motivate both modeling and algorithmic research activities, and its own place in the knapsack literature (Cacchiani et al., 2022b). MMdKFSP can be viewed as a variant of the more established MKP and MdKP. Reviews on these problems can be found in Chekuri and Khanna (2000), Dell'Amico et al. (2019), Ferreira et al. (1996), Fréville (2004), and Pisinger (1999). Other KP variants related to the problem under study include the cases in which the items are *partitioned* or *grouped* into different families or classes, introducing related adjustments in the model constraints and/or the objective functions. A first example is given by the KP with Setups (KPS), where the set of items is subdivided into a set of families. In this case, unlike MMdKFSP, a single item can be chosen only if a *setup* cost is paid for the family containing it (McLay and Jacobson, 2007; Michel et al., 2009; Della Croce et al., 2017; Furini et al., 2018; Pferschy and Scatamacchia, 2018; Amiri, 2020; Amiri and Barkhi, 2021). Differently, the Fixed-Charge Multiple Knapsack Problem (FCMKP) has been proposed by Yamada and Takeoka (2009) as an extension of the MKP in which a specific fixed cost must be paid for each knapsack used in the solution. The problem is to decide the set of knapsacks to use, and to assign items to them, so that the total net profit (item profits minus knapsack costs) is maximized.

Another case arising in various application contexts is the Class-Constrained Multiple Knapsack (CCMK) (Shachnai and Tamir, 2001), having as objective the maximization of the total profit

obtained by the chosen items, in which sets of items of different profits and sizes are included into different *classes*, and each knapsack is characterized by a bounded capacity. Moreover, in CCMK, an additional constraint is considered on the number of different classes of items each knapsack can host.

Regarding the *multi-objective* version of the MdKP and the correspondent optimization methods, a comprehensive review is offered in Lust and Teghem (2012), while Ceselli and Righini (2006) and Della Croce et al. (2019) addressed a *Penalized* version of KP (PKP). In the latter problem, each single item is characterized by a *penalty* of constant value, and the objective is modeled by a function representing the total profit that is reduced by the most significant penalty shown by the chosen items.

A further relevant problem related to MMdKFSP is the Multiple-choice Multidimensional Knapsack Problem (McMKP) (Nauss, 1978). In this case, the set of items available for selection is divided into several groups and various types of capacitated resources, each item generates a specific profit contributing to the overall objective, and needs a certain amount of each resource. The goal of the McMKP is to choose exactly one item from each group to maximize the total profit of the selection, while the utilization of each resource has to respect the limit expressed by a set of knapsack constraints. Recently, due to its theoretical and practical relevance (Mansi et al., 2013; Shojaei et al., 2013), the McMKP is receiving increasing attention from researchers, also justified by various applications (Hifi et al., 2004; Han et al., 2010; Chen and Hao, 2014; Mansini and Zanotti, 2020).

Other relevant cases close to MMdKFSP are some extensions of MKP appearing in *assignment* or *packing* models, more specifically the Multiple Knapsack Assignment Problem (MKAP), the All-or-Nothing Generalized Assignment (AGAP), and the problem of packing groups of items in multiple knapsacks (GMKP). Like in MMdKFSP, in MKAP the items available for allocation in the knapsacks are partitioned in sets, but in this case each knapsack may only receive items from one of the sets induced by the partition (Kataoka and Yamada, 2014; Martello and Monaci, 2020). Also the AGAP (Adany et al., 2013) considers items partitioned into different groups, each item has its own size and a payoff, however the latter also depends on the knapsack to which it is assigned. The overall size of the items assigned to a knapsack, as usual, is subject to a capacity constraint. Furthermore, in this problem, each knapsack is able to allocate at most one item from each group introduced by the partition. A group of items is considered satisfied only when all its own items are assigned to a knapsack. The global objective is to find a feasible packing of a subset of items in the knapsacks maximizing the total profit given by the satisfied groups. More similar to MMdKFSP seems the case of packing Groups of Items into Multiple Knapsacks (GMKP), recently studied in Chen and Zhang (2018). Also this problem presents items partitioned in different groups. Each single item has its own size, but the utility is associated with the groups instead of the items. The utility of a group can be obtained only when all items included in it are assigned. They can be allocated in different knapsacks, but the latter are *identical* and, more relevant, the model does not consider penalties when splitting groups.

## 3. The MMdKFSP problem description

This section gives a formal description of the optimization problem considered in this paper. The set $\mathcal{I}$ contains $n = |\mathcal{I}|$ given items. A generic item is denoted by $i \in \mathcal{I}$, which takes values in $\{1 \ldots, n\}$.

Table 1
Explanation of the parameters and variables used in the model

| Parameter | Description |
|---|---|
| $\mathcal{I}$ | Set of items available for selection; $\lvert\mathcal{I}\rvert = n$ |
| $\mathcal{F}$ | Set of item families; $\lvert\mathcal{F}\rvert = m$ |
| $\mathcal{F}_j$ | Set of items included in the $j$th family; $j \in \{1, \ldots, m\}$ |
| $\mathcal{K}$ | Set of available knapsacks; $\lvert\mathcal{K}\rvert > 1$ |
| $D$ | Number of relevant dimensions of knapsacks |
| $C_k^d$ | Capacity value of knapsack $k$ concerning dimension $d \in \{1, \ldots, D\}$ |
| $c_i^d$ | Demand of resources of item $i \in \mathcal{I}$ concerning dimension $d \in \{1, \ldots, D\}$ |
| $p_j$ | Profit of family $\mathcal{F}_j$ |
| $\delta_j$ | Split-penalty of family $\mathcal{F}_j$ |
| **Variable** | **Description** |
| $x_j$ | (0/1) variable having value 1 if and only if family $\mathcal{F}_j$ is selected |
| $y_{ik}$ | (0/1) variable having value 1 if and only if item $i$ is allocated in knapsack $k$ |
| $z_{jk}$ | (0/1) variable having value 1 if and only if at least one item included in family $\mathcal{F}_j$ is allocated in knapsack $k$ |
| $u_j$ | (0/1) variable assuming value 1 if family $j$ is split into two or more knapsacks and value 0 if all items included in family $\mathcal{F}_j$ are allocated in the same knapsack or if the family has not been selected |

Items in $\mathcal{I}$ are partitioned into $m$ families indicated as $\mathcal{F}_j \subset \mathcal{I}$, with $j \in \{1, \ldots, m\}$. Hence, each single item is included in one (and only one) family (i.e., $\mathcal{F}_h \cap \mathcal{F}_k = \emptyset,\ \forall h \neq k$, and $\cup_{j=1}^{m} \mathcal{F}_j = \mathcal{I}$). Hereinafter, $\mathcal{F}$ indicates the set of families. A set of heterogeneous multidimensional knapsacks, $\mathcal{K}$ is given. Each knapsack $k \in \mathcal{K}$ has a number $D$ of relevant dimensions, and specific capacity values $C_k^d$ representing the amount of available resources for each dimension $d \in \{1, \ldots, D\}$. Each item $i$ is associated with a demand of resources, referred to as $c_i^d$ with $d \in \{1, \ldots, D\}$, and $i \in \mathcal{I}$, respectively, and the global requirement of a resource for a family corresponds to the sum of all the requirements of the items included in it.

Each family $\mathcal{F}_j$ has associated a specific profit $p_j$, assumed to be a deterministic and constant value, which is known before optimization takes place. The profit $p_j$ is earned only if the corresponding family $\mathcal{F}_j$ is selected, that is, if all the items included in $\mathcal{F}_j$ are allocated in some $k \in \mathcal{K}$. Items of the same family can be allocated to the same knapsack or to different knapsacks. In the second case, a family-dependent penalty cost indicated as $\delta_j$ is applied. The overall goal is to maximize the total *utility* given by the profits obtained by allocating the families, net of split penalties.

Section 3.1 presents a mathematical programming formulation of this problem, whereas in Section 3.2 a method to obtain upper bounds based on a specific problem relaxation is introduced.

### 3.1. Mathematical programming formulation

This section introduces the mathematical model adopted for the MMdKFSP, which can be formulated as an Integer Linear Optimization Problem indicated as ILP_MMdKFSP in what follows. To make the model description clearer, Table 1 summarizes the used parameters and variables.

The general goal of MMdKFSP is to maximize the profit obtained by allocating the selected items, discounting the penalties paid for splitting families across two or more knapsacks, as stated in the objective function (1). For clarity, the two terms are kept separate. The (0/1) variable $x_j$ takes value 1 if and only if family $\mathcal{F}_j$ is selected, whereas the (0/1) variable $u_j$ is set to 0 when all items of family $\mathcal{F}_j$ are allocated in the same knapsack or the family has not been selected, and has value 1 when there is a splitting.

$$\text{(ILP\_MMdKFSP)} \qquad \max \sum_{j|\mathcal{F}_j \in \mathcal{F}} p_j x_j - \sum_{j|\mathcal{F}_j \in \mathcal{F}} \delta_j u_j \tag{1}$$

Subject to:

$$\sum_{k \in \mathcal{K}} y_{ik} = x_j \quad \forall i \in \mathcal{I}, \quad \forall j | i \in \mathcal{F}_j, \tag{2}$$

$$\sum_{i \in I} c_i^d \, y_{ik} \leq C_k^d \quad \forall k \in \mathcal{K}, \quad \forall d \in \{1, \ldots, D\}, \tag{3}$$

$$z_{jk} \geq \frac{1}{|\mathcal{F}_j|} \sum_{i \in \mathcal{F}_j} y_{ik} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}, \tag{4}$$

$$u_j \geq \frac{1}{|\mathcal{K}| - 1} \left( \sum_{k \in \mathcal{K}} z_{jk} - 1 \right) \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \tag{5}$$

$$x_j, u_j \in \{0, 1\} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \tag{6}$$

$$z_{jk} \in \{0, 1\} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}, \tag{7}$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in \mathcal{I}, \quad \forall k \in \mathcal{K}. \tag{8}$$

Constraints (2) make sure that if a family is chosen, all the items included in it are allocated in some knapsack. We can note that $y_{ik}$ is a (0/1) variable having value 1 if and only if item $i$ is allocated in knapsack $k$. The group of Inequalities (3) implies for all knapsacks the respect of the capacity constraints for all the considered dimensions. The set of Constraints (4) identifies whether a knapsack contains items of a given family, whereas Constraints (5) enable the formulation to detect if a family is allocated in a single knapsack or split into two or more of them. The (0/1) variable $z_{jk}$ assumes the value 1 only if at least one item included in the family $\mathcal{F}_j$ is allocated in the knapsack $k$. Finally, Constraints (6)–(8) define the domain of the variables.

### *3.2. Upper bounds through multi-knapsack relaxation*

To provide tight upper bounds for the MMdKFSP, we propose a Multi-Knapsack relaxation (MKr) that keeps unchanged the number and the capacity of the knapsacks of the original problem while

treating each family $\mathcal{F}_j$ as a single compound item $S_j$ having a demand for each resource type given by the sum of the requirements of all items included in it, that is, $c_j^d = \sum_{i \in \mathcal{F}_j} c_i^d$.

The peculiarity of this approach is that each item $S_j$ can be split into arbitrary portions $Q_{jk}$ (with $0 \leq Q_{jk} \leq 1$, and $\sum_{k \in \mathcal{K}} Q_{jk} = 1$) that can be assigned to different knapsacks $k \in \mathcal{K}$, and if it is the case, a split penalty is charged. Since the number of items is sensibly smaller than in the original problem, this problem can be quickly solved to optimality.

The advantage of the proposed relaxation mainly lies in the inclusion of the splitting penalties in the objective function of the relaxed problem to obtain a tight upper bound, especially in those cases where, due to the difference in size between families and knapsacks, the number of split families is quite large.

The integer linear programming formulation for MKr is reported in the following:

$$\text{(MKr)} \qquad \max \sum_{j|\mathcal{F}_j \in \mathcal{F}} p_j x_j - \sum_{j|\mathcal{F}_j \in \mathcal{F}} \delta_j u_j \tag{9}$$

Subject to:

$$\sum_{k \in \mathcal{K}} Q_{jk} = x_j \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \tag{10}$$

$$\sum_{j|\mathcal{F}_j \in \mathcal{F}} c_j^d Q_{jk} \leq C_k^d \quad \forall k \in \mathcal{K}, \quad \forall d \in \{1, \ldots, D\}, \tag{11}$$

$$z_{jk} \geq Q_{jk} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}, \tag{12}$$

$$u_j \geq \frac{1}{|\mathcal{K}| - 1} \left( \sum_{k \in \mathcal{K}} z_{jk} - 1 \right) \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \tag{13}$$

$$x_j, u_j \in \{0, 1\} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \tag{14}$$

$$z_{jk} \in \{0, 1\} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}, \tag{15}$$

$$0 \leq Q_{jk} \leq 1 \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}. \tag{16}$$

The objective function of MKr is reported in (9). Constraints (10) imply that if a family $j \in \mathcal{F}$ is selected (i.e., $x_j = 1$), it must be fully assigned to one or more knapsacks, in arbitrary shares. If a family is not selected, no portion of it is assigned to knapsacks. Capacity constraints for each resource must be respected for all the knapsacks, as imposed by Constraints (11). Constraints (12) allow the formulation to identify whether a portion of a family has been assigned to a knapsack or not, whereas Constraints (13) detect whether a family is split or not. Finally, Constraints (14)–(16) define the domain of the variables.

## 4. Combinatorial benders cuts

Benders Decomposition (BD) (Benders, 1962) is an established methodology based on the main idea of decomposing the optimization problem under study into a Main Problem (MP) and a Sub-Problem (SP) that are sequentially solved in an in iterative fashion.

A variant of the classical BD method, referred to as Logic Based Benders Decomposition (LBBD), has been proposed by Hooker and Ottoson (2003). In the approach proposed by the authors, MP considers only variables contributing to the objective function, while the remaining ones, responsible only for feasibility, are addressed in SP. This way, the SP turns out to be a pure feasibility problem. Whether SP results infeasible, cuts are added to MP to exclude from MP's search space solutions detected as infeasible by SP. As soon as SP provides a feasible solution, this solution is proved to be optimal for OP, too.

A particular case of LBBD, named Combinatorial Benders Cuts based approach, (CBC), has been introduced by Codato and Fischetti (2006). This approach is specifically designed for models dealing with binary variables tied together by a large number of logical implications. In such a scenario, every time a realization of MP variables yields an infeasible SP, a CBC, forcing to 0 at least one of the variables with value 1, is included in the MP. Such a procedure allows the formulation to exclude from the search space a single solution, which was proved to be infeasible, at a time. This may yield, in cases of highly constrained problems presenting large infeasible portions of the search space, a very slow convergence toward an optimal solution.

Stronger cuts can be earned by detecting, optimally or heuristically, a subset of variables that are responsible for infeasibility. Such cuts can be considered *stronger* because they allow the model to exclude, from MP search space, several solutions at a time, cutting-off large sub-spaces, strongly speeding up the convergence toward an optimal solution, as in the recently published papers (Bruglieri et al., 2021; Mancini and Gansterer, 2021), in which different CBC approaches for vehicle routing problems are provided.

It is worth remarking that the convergence of the algorithm is guaranteed with both standard and enhanced cuts, in a finite number of iterations (Chu and Xia, 2004; Codato and Fischetti, 2006). In fact, in the worst case a number of iterations equal to the number of possible realizations of the variable set, minus one is required. However, this number can become huge even for problems involving a reduced number of variables, since it grows exponentially with the number of variables. It is important to point out that, given a MIP model, different decomposition strategies, leading to different MPs, may show completely different performances in terms of convergence speed, since both the number of iterations required and the computational time required to solve each of them can vary sensibly. Therefore, the definition of MP (and consequently of SP) plays a crucial role in the *success* of the proposed method and therefore this issue must be carefully addressed. Nevertheless, also the effectiveness of the proposed combinatorial Benders Cuts plays a fundamental role. In fact, classical *no good cuts*, which only exclude from the solution space one solution at a time, may be *weak* and require a very large number iterations to converge to the optimal solution. Conversely, the adoption of *stronger* cuts that allow the model to exclude large portions of the solution space simultaneously, and consequently an exponential number of solutions, would speed up the convergence of the algorithm, significantly reducing the number of iterations required.

### 4.1. A novel combinatorial benders cuts algorithm for the MMdKFSP

The novel CBC we propose aims at exploiting in the MP the problem specific MKr relaxation introduced in Section 3.2; this method is denoted by CBC-MKr. In the MKr relaxation, each family is considered as a compound item having as demands, for each resource, the sum of the demands of all the items belonging to it. Each family can be split into portions of arbitrary size, if needed to fit the knapsacks but, if it happens, a split penalty is charged. In this case, the objective value of MP optimal solution exactly corresponds to that of MMdKFSP and thus the obtained solution is also feasible for it. Therefore, the SP is only responsible for checking feasibility. If the optimal solution of MP is feasible also for MMdKFSP, then it is automatically optimal for it.

When the CBC-MKr algorithm starts, MP exactly corresponds to the MKr relaxation, whereas SP is derived by the original formulation, ILP_MMdKFSP, by setting the value of the $x_j$ variables to 1 if they have been selected in the optimal solution of the MP, $x_j = 0$ otherwise. The method works as follows. At each iteration, first MP is solved, then the set of families selected in the optimal solution are forced to be selected also in SP, which is solved in turn. If SP is feasible, then the optimal solution of MP is certified optimal also for MMdKFSP. Conversely, if SP is infeasible, a CBC is added to MP to exclude a future simultaneous selection of all the families chosen at the previous iteration. These steps are reiterated until the SP feasibility is reached.

The rationale of this approach lies in the design of a relaxation able to provide a tight upper bound of the optimal solution, with the aim to reduce the number of iterations and cuts to add to MP to close the problem to optimality, with a consequent boost on computational times. It is worth noting that the adopted cuts allow us to simultaneously exclude a large number of solutions from the MP solution space. Indeed, excluding a combination of selected families, we are removing all the solutions in which those families are selected, whichever their assignment to knapsacks is. The number of possible different combinations of family assignments to knapsacks, given a fixed set of families, exponentially grows with the size of this set.

This method guarantees convergence toward the optimal solution in a finite number of iterations. In fact, the number of possible combinations of selected families is exponential but it is finite and in the worst case, the number of iterations needed is at most equal to $2^{|\mathcal{F}|}$. However, the tighter is the upper bound offered by the relaxed problem MKr, the lower is the number of required iterations.

At a generic iteration $h$ of the CBC-MKr procedure, the Main Problem $\text{MP}^\text{h}$ and the Sub-Problem $\text{SP}^\text{h}$ are solved in sequence. Once an optimal solution $\bar{x} = \{\bar{x}_1^h, \bar{x}_2^h, \ldots, \bar{x}_m^h\}$ of value $W_{MP^h}$ for $\text{MP}^\text{h}$ is calculated, the related set of selected families is tested for feasibility for the original ILP_MMdKFSP. To this aim, $\text{SP}^\text{h}$ solves the ILP_MMdKFSP with an additional set of constraints to fix the values of the $x_j$ variables to those provided by the optimal solution of $\text{MP}^\text{h}$:

$$x_j = \bar{x}_j^h \quad \forall j | \mathcal{F}_j \in \mathcal{F}. \tag{17}$$

If $\text{SP}^\text{h}$ is infeasible, then the following cut is introduced to prevent the MP from including the same subset of families in another solution of future iterations:

$$\sum_{j|\bar{x}_j^h=1} x_j \leq \alpha^h - 1, \tag{18}$$

where $\alpha^h = \sum_{j|\mathcal{F}_j \in \mathcal{F}} \bar{x}_j^h$.

**Algorithm 1.** CBC-MKr Algorithm for MMdKFSP

**Input :** $MMdKFSP$; // MMdKFSP instance
**Output:** $Solution$; // Optimal solution for MMdKFSP instance
$W$; // Optimal objective function value

$h = 1,\ Optim\_Phase = TRUE,\ Solution = \emptyset$
MP$^h$ = *get_MKr_formulation*(MMdKFSP)
**while** $Optim\_Phase$ **do**
  $\bar{x}^h$ = *solve*(MP$^h$)
  $W_{MP^h}$ = *optValue*(MP$^h$, $\bar{x}^h$)
  SP$^h$ = MMdKFSP $\wedge$ $(x_j = \bar{x}_j^h \quad \forall j|\mathcal{F}_j \in \mathcal{F})$
  $\bar{s}^h$ = *solve*(SP$^h$)
  $W_{SP^h}$ = *optValue*(SP$^h$, $\bar{s}^h$)
  **if** *isFeasible*(*SP*$^h$, $\bar{s}^h$) **then**
    $Solution = \bar{s}^h$
    $W = W_{SP^h}$
    $Optim\_Phase = FALSE$
  **else**
    $\alpha^h = \sum_{j|\mathcal{F}_j \in \mathcal{F}} \bar{x}_j^h$
    MP$^{h+1}$ = MP$^h$ $\wedge \sum_{j|\bar{x}_j^h = 1} x_j \leq \alpha^h - 1$
  $h = h + 1$
**return** $Solution, W$

On the basis of the MKr mathematical formulation introduced in Section 3.2, and the Benders cuts defined in (18), the integer linear program MP$^{\mathrm{h}}$ for the Main Problem MP at the generic iteration $h$ can be formulated as reported from (19) to (27). Note that the set of Constraints (24) collects in MP$^{\mathrm{h}}$ all the Benders cuts (18) generated up to the iteration $h$.

To summarize the structure of the CBC-MKr algorithm for MMdKFSP, a detailed pseudo-code is reported in the following, where $W_{\Gamma}^{h}$ indicates the value of the optimal solution for a problem $\Gamma = \{\mathrm{MP}^{\mathrm{h}}, \mathrm{SP}^{\mathrm{h}}\}$ at iteration h, respectively (Algorithm 1).

$$(\mathrm{MP}^{\mathrm{h}}) \qquad \max \sum_{j|\mathcal{F}_j \in \mathcal{F}} p_j x_j - \sum_{j|\mathcal{F}_j \in \mathcal{F}} \delta_j u_j \tag{19}$$

Subject to:

$$\sum_{k \in \mathcal{K}} Q_{jk} = x_j \quad \forall j|\mathcal{F}_j \in \mathcal{F}, \tag{20}$$

$$\sum_{j|\mathcal{F}_j \in \mathcal{F}} c_{\hat{j}}^{d} Q_{jk} \leq C_k^d \quad \forall k \in \mathcal{K}, \quad \forall d \in \{1, \ldots, D\}, \tag{21}$$

$$z_{jk} \geq Q_{jk} \quad \forall j|\mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}, \tag{22}$$

$$u_j \geq \frac{1}{|\mathcal{K}| - 1} \left( \sum_{k \in \mathcal{K}} z_{jk} - 1 \right) \quad \forall j|\mathcal{F}_j \in \mathcal{F}, \tag{23}$$

$$\sum_{j|\bar{x}_j^{h-1}=1} x_j \leq \alpha^{h-1} - 1 \quad \forall h \geq 2 \mid \text{SP}^{\text{h-1}} \text{ infeasible}, \tag{24}$$

$$x_j, u_j \in \{0, 1\} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \tag{25}$$

$$z_{jk} \in \{0, 1\} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}, \tag{26}$$

$$0 \leq Q_{jk} \leq 1 \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}. \tag{27}$$

## 5. Computational analysis

In this section, we describe the computational experiments performed to analyze the behavior of the proposed algorithm and evaluate its effectiveness. The proposed algorithm is compared against state-of-the-art alternative approaches both in terms of execution times and the quality of the results. More specifically, to assess the performance of the CBC-MKr approach proposed in this paper, we compare it against the baseline ILP formulation presented in Section 3.1 (ILP_MMdKFSP) and the state-of-the-art Combinatorial Benders Cuts method, hereinafter referred to as CBC-SKr, proposed in Mancini et al. (2021) in which the MP consists in a Single-Knapsack relaxation (indicated as SKr).

The computational experiments presented in this article have been run on a system with a 2.4 GHz Intel-i7-5500U CPU, with 16GB of RAM and Windows 7 OS. All the (single-threaded) procedures have been implemented in the Xpress-mosel language. Importantly, all ILP models in this experimental study campaign, both those underpinning CBC-MKr and those of other approaches, have been solved with the same solver, namely Xpress 7.9. All reported computation times for the execution of the algorithms are expressed in seconds.

The rest of this section is organized as follows. Section 5.1 illustrates the sets of test instances considered in our computational experiments. They include both benchmark instances available from the recent literature, and a set of new instances designed to better investigate the behavior of the proposed algorithm, providing a more comprehensive experimental setting. The successive Section 5.2 reports, analyzes, and discusses the obtained results. Then, the application of the proposed method to the more general Fixed-Charge MMdKFSP case is also considered.

### 5.1. Experimental setup

In order to evaluate the performance of the proposed approach, a benchmark consisting of four instance sets from the literature has been exploited. In addition, a new set of test instances have been generated, extending the experimental settings toward more challenging cases. The benchmark instances available in the literature, in what follows referred to as MCM2019, have been proposed in Mancini et al. (2021), and are composed of four groups of cases, indicated as $Gi$, with $i \in \{1, \ldots, 4\}$. The first group, $G1$, contains 360 bi-dimensional instances (i.e., $D = 2$) having the number of

knapsacks $|\mathcal{K}|$ ranging from 3 to 10, the number of items $n$ is between 400 and 600, while the number of families $m$ assumes values from 6 to 13. The instances are organized into four scenarios to consider various capacity settings characterized by patterns based on different $C_k^d$ values and their distribution among knapsacks. The second group of test instances, $G2$, has been proposed to investigate the scalability of algorithms when knapsacks with several dimensions are considered. In $G2$ instances, all knapsacks ($|\mathcal{K}| = 10$) and items ($n = 600$) have the same number of dimensions $D \in \{2, 4, 6, 8\}$.

As for the other parameters, in the groups $G2$-$G4$ the general scheme of the instances of $G1$ adopts an uneven arrangement of the knapsack capacities, with one sensibly more capacious than the others. To this aim, a parameter $\beta_{kd}$ has been introduced to represent the filling ratio of knapsack $k$ with respect to the $d$th dimension. In these instances, the capacity of first knapsack (indicated with $k = 1$) for a certain dimension $d$ is equal to the maximum of the per-family consolidated demand of $d$, that is

$$C_1^d = \max_{j|\mathcal{F}_j \in \mathcal{F}} \sum_{i \in \mathcal{F}_j} c_i^d \qquad \forall d \in \{1, \ldots, D\}, \tag{28}$$

where $c_i^d$ represents for each item $i \in \mathcal{F}_j$ the demand of resource $d \in \{1 \ldots D\}$. As for the other knapsacks, the capacity for each dimension $d$ has been calculated with the following formula:

$$C_k^d = \frac{\beta_{kd}}{|\mathcal{K}|} \sum_{i \in \mathcal{I}} c_i^d \quad \forall k \in \mathcal{K}, k \neq 1, \forall d \in \{1, \ldots, D\} \tag{29}$$

with $\beta_{kd}$ drawn from the uniform distribution $\overline{U}(0.27, 0.33)$.

For each value of $D$, 10 instances are considered in $G2$, making a total of 40 cases. The group $G3$ has been proposed to study the influence of the number of knapsacks on the algorithms' performance, and contains 30 bi-dimensional instances with $|\mathcal{K}| \in \{10, 15, 20\}$, whereas the number of items is fixed at $n = 600$. Finally, the group of instances $G4$ in the MCM2019 set has been proposed to investigate the algorithmic performance for a growing number of items $n$. This group includes 30 bi-dimensional cases with $n \in \{600, 800, 1000\}$ and $|\mathcal{K}| = 10$.

However, in all $G1$-$G4$ groups the average size of the items is very small compared to the knapsacks' capacity, producing a granularity effect. Consequently, in these cases, considering each family as a single item is a very good approximation. Indeed, since the items are small, it is very likely that, given a selection of families, a feasible partition of items into knapsacks exists.

This idea motivated the use of SKr in which all knapsacks are collapsed into a fictitious macro-knapsack, $\hat{k}$, whose capacity, for each resource, corresponds to the sum of the capacity of all the knapsacks, $C_{\hat{k}}^d = \sum_k C_k^d$. In this scheme, each family is considered as a single item having, for each resource, a demand given by the sum of the requirements of the items included.

In order to study more challenging scenarios, in this paper we introduce a new set of bi-dimensional instances, named $G5$, in which the item size (in terms of demand of resources $c_i^d$, with $i \in \mathcal{I}$ and $d \in \{1, \ldots, D\}$) is relatively large respect to the capacity of the knapsacks ($C_k^d, k \in \mathcal{K}$) associated to dimensions $d \in \{1, \ldots, D\}$, that is, each knapsack can contain a reduced number of items. The structure of the instances of $G5$ follows the uneven pattern, with one of the knapsacks significantly wider in size than the others, as described in (28) and (29). In addition to the

Table 2
Aggregated results for the test instances MCM2019

| | ILP_MMdKFSP | | CBC-SKr | | | | CBC-MKr | | |
|---|---|---|---|---|---|---|---|---|---|
| Set (#Inst) | %Os | T(s) | %Os | T(s) | Fc | Pc | %Os | T(s) | C |
| G1 (360) | 98.3 | 104.90 | 99.7 | 87.07 | 0.02 | 4.61 | 100.00 | 0.14 | 0.00 |
| G2 (40) | 90.0 | 908.99 | 100.0 | 318.36 | 3.20 | 29.20 | 100.00 | 4.73 | 0.60 |
| G3 (30) | 83.3 | 1056.36 | 86.7 | 795.77 | 0.33 | 18.10 | 100.00 | 0.62 | 0.00 |
| G4 (30) | 60.0 | 2026.33 | 70.0 | 1695.07 | 0.73 | 69.63 | 100.00 | 0.58 | 0.00 |
| **Avg.** | **94.1** | **362.18** | **96.9** | **228.27** | **0.36** | **11.87** | **100.00** | **0.77** | **0.07** |

benchmark sets form the literature, these instances allow us to investigate the impact of the number of families. They are expected to be challenging since, given a set of selected families, a feasible partition of items into knapsacks may be difficult to find or not exist at all. The set *G5* consists of four groups (indicated as $G5_\gamma$, with $\gamma = 1, \ldots, 4$), each containing 10 instances. The number of knapsacks $|\mathcal{K}| = 10$, the number of items $n = 100$, and the number of dimensions $D = 2$ is the same in all groups, while the number of families $m$ increases gradually from $G5_1$ to $G5_4$. More in detail, in the instances of groups $G5_\gamma$ the number of families is generated from a distribution characterized by $\mathbb{E}(D) = 5\gamma$.

The overall number of instances investigated in our computational experiments amounts to 500.

### 5.2. Computational results and discussion

This section reports on the results obtained testing the CBC-MKr algorithm presented in Section 4.1 on the just-introduced benchmark instances, also conducting a comparison with alternative models and methods available from the literature.

The computational experiments have been carried out running all considered methods on each of the 500 instances included in the benchmark sets allowing a maximum computation time of 3600 s for each run (i.e., an algorithm is allowed to run at most one hour on each instance), and recording the objective value of the best solution found together with information on its optimality status.

Table 2 shows the results related to the instances MCM2019, while the results concerning the instances of the set *G5* proposed in this paper are detailed in Tables 2–6. In order to assess the performance of the CBC-MKr approach proposed in this paper, we compare it against CBC-SKr and the baseline ILP formulation presented in Section 3.1 (ILP_MMdKFSP) running on the same computational setting on the instances of the *G*1-*G*4 group.

Each row of this table shows the average results, by groups, achieved by the three approaches considered. The first column lists the group identifier together with the number of instances included in the group (Set (#Inst)). In addition, the last row presents the averaged results on all groups of instances.

The table is divided into three sections, each one dedicated to summarize the results of one of the methods considered in this experiment. These sections are identified with the labels ILP_MMdKFSP, CBC-SKr, and CBC-MKr, respectively.

Table 3
Computational results for the test instances $G5_1$. For all instances $n = 100$ and $|\mathcal{K}| = 10$

| | ILP_MMdKFSP | | CBC-SKr | | | | CBC-MKr | | |
|---|---|---|---|---|---|---|---|---|---|
| Inst (m) | OF | T(s) | OF | T(s) | Fc | Pc | OF | T(s) | C |
| 1 (5) | **206** | 1.76 | **206** | 2.81 | 0 | 4 | **206** | 0.29 | 0 |
| 2 (6) | **115** | 1.05 | **115** | 1.87 | 0 | 4 | **115** | 0.03 | 0 |
| 3 (6) | **236** | 2.67 | **236** | 12.31 | 0 | 4 | **236** | 0.29 | 0 |
| 4 (6) | **248** | 2.97 | **248** | 18.05 | 0 | 13 | **248** | 0.30 | 0 |
| 5 (6) | **250** | 1.27 | **250** | 2.60 | 0 | 3 | **250** | 0.06 | 0 |
| 6 (5) | **145** | 6.07 | **145** | 10.29 | 0 | 4 | **145** | 0.33 | 0 |
| 7 (5) | **204** | 1.42 | **204** | 0.65 | 0 | 2 | **204** | 0.05 | 0 |
| 8 (5) | **165** | 1.82 | **165** | 2.96 | 0 | 4 | **165** | 0.17 | 0 |
| 9 (5) | **165** | 2.12 | **165** | 5.52 | 0 | 6 | **165** | 0.24 | 0 |
| 10 (5) | **126** | 1.10 | **126** | 1.86 | 0 | 5 | **126** | 0.22 | 0 |
| **Avg.** | **186.00** | **2.22** | **186.00** | **5.89** | **0.00** | **4.90** | **186.00** | **0.20** | **0.00** |

Table 4
Computational results for the test instances $G5_2$. For all instances $n = 100$ and $|\mathcal{K}| = 10$

| | ILP_MMdKFSP | | CBC-SKr | | | | CBC-MKr | | |
|---|---|---|---|---|---|---|---|---|---|
| Inst (m) | OF | T(s) | OF | T(s) | Fc | Pc | OF | T(s) | C |
| 1 (11) | **230** | 8.25 | **230** | 360.00 | 26 | 69 | **230** | 0.53 | 0 |
| 2 (10) | **243** | 6.20 | **243** | 106.77 | 6 | 42 | **243** | 0.48 | 0 |
| 3 (9) | **216** | 2.16 | **216** | 3.59 | 0 | 8 | **216** | 0.32 | 0 |
| 4 (10) | **217** | 4.34 | **217** | 59.40 | 4 | 20 | **217** | 0.47 | 0 |
| 5 (11) | **215** | 9.32 | **215** | 187.93 | 11 | 47 | **215** | 0.47 | 0 |
| 6 (14) | 426 | 3600 | 426 | 3600 | 59 | 507 | **428** | 11.27 | 1 |
| 7 (15) | 452 | 3600 | 452 | 3600 | 80 | 321 | **452** | 10.58 | 1 |
| 8 (15) | 513 | 3600 | **532** | 3513.50 | 68 | 455 | **532** | 0.91 | 0 |
| 9 (15) | **517** | 1796.53 | 517 | 3600 | 86 | 455 | **517** | 0.78 | 0 |
| 10 (15) | 427 | 3600 | 424 | 3600 | 106 | 342 | **427** | 0.72 | 0 |
| **Avg.** | **345.60** | **1622.68** | **347.20** | **1863,12** | **44.60** | **226.60** | **347.70** | **2.64** | **0.20** |

The section concerning the ILP baseline outcomes features two columns. The first column (%Os) shows the percentage of instances of each group solved to optimality; the second one (T(s)) reports the average calculation time (in seconds). Note that for this algorithm, as for the other methods involved in this comparison, a timeout of 60 minutes per single instance has been set.

The results of the CBC-SKr algorithm are discussed in the second section of the table. However, the scheme of this sub-table consists of four columns, the first two of which have the same meaning as those in the first section (i.e., percentage of instances resolved to optimality and average execution time). The remaining two columns (Fc) and (Pc) describe the behavior of the algorithm in terms of the average number of occurred feasibility and penalty cuts added per instance on average, respectively.

In fact, in the CBC-SKr method the MP solution determines an optimal selection of families while SP plays a twofold role. First, it checks whether the optimal solution of the MP is feasible for MMdKFSP, and second, it detects if split penalties occur. When the solution of the MP turns

Table 5
Computational results for the test instances $G5_3$. For all instances $n = 100$ and $|\mathcal{K}| = 10$

| | ILP_MMdKFSP | | CBC-SKr | | | | CBC-MKr | | |
|---|---|---|---|---|---|---|---|---|---|
| Inst (m) | OF | T(s) | OF | T(s) | Fc | Pc | OF | T(s) | C |
| 1 (15) | 469 | 3600 | 465 | 3600 | 71 | 347 | **470** | 12.65 | 1 |
| 2 (14) | **328** | 122.48 | 328 | 3600 | 67 | 603 | **328** | 0.57 | 0 |
| 3 (15) | 472 | 3600 | 477 | 3600 | 93 | 284 | **484** | 23.61 | 2 |
| 4 (14) | **398** | 448.75 | 398 | 3600 | 92 | 597 | **398** | 0.35 | 0 |
| 5 (14) | 376 | 3600 | **376** | 2989.66 | 110 | 509 | **376** | 0.32 | 0 |
| 6 (11) | 220 | 3600 | **226** | 159.04 | 11 | 39 | **226** | 15.19 | 1 |
| 7 (11) | **284** | 6.24 | **284** | 75.90 | 2 | 19 | **284** | 0.42 | 0 |
| 8 (12) | **318** | 8.98 | **318** | 273.36 | 17 | 57 | **318** | 0.37 | 0 |
| 9 (11) | 318 | 3600 | **328** | 258.82 | 16 | 59 | **328** | 16.45 | 1 |
| 10 (10) | **170** | 14.37 | **170** | 40.45 | 1 | 13 | **170** | 0.38 | 0 |
| **Avg.** | **335.3** | **1860.08** | **337** | **1819.72** | **48.00** | **252.70** | **338.20** | **7.03** | **0.50** |

Table 6
Computational results for the test instances $G5_4$. For all instances $n = 100$ and $|\mathcal{K}| = 10$

| | ILP_MMdKFSP | | CBC-SKr | | | | CBC-MKr | | |
|---|---|---|---|---|---|---|---|---|---|
| Inst (m) | OF | T(s) | OF | T(s) | Fc | Pc | OF | T(s) | C |
| 1 (20) | **730** | 471.02 | 721 | 3600 | 194 | 195 | **730** | 2.25 | 0 |
| 2 (20) | 596 | 3600 | 595 | 3600 | 190 | 213 | **604** | 2.01 | 0 |
| 3 (21) | **718** | 495.13 | 682 | 3600 | 211 | 159 | **718** | 1.54 | 0 |
| 4 (19) | **552** | 512.29 | 536 | 3600 | 176 | 284 | **552** | 0.66 | 0 |
| 5 (20) | 585 | 3600 | 585 | 3600 | 193 | 173 | **595** | 4.69 | 0 |
| 6 (20) | 761 | 3600 | 757 | 3600 | 160 | 205 | **772** | 1.62 | 0 |
| 7 (20) | 568 | 3600 | 568 | 3600 | 131 | 300 | **572** | 1.35 | 0 |
| 8 (21) | 645 | 3600 | 644 | 3600 | 168 | 197 | **649** | 1.80 | 0 |
| 9 (20) | **622** | 145.49 | 608 | 3600 | 122 | 251 | **622** | 0.55 | 0 |
| 10 (20) | **738** | 1015.58 | 729 | 3600 | 179 | 202 | **738** | 1.03 | 0 |
| **Avg.** | **651.50** | **2063.95** | **642.50** | **3600** | **172.40** | **217.90** | **655.20** | **1.75** | **0.00** |

out to be infeasible, a *Feasibility Cut* (Fc) is added to the MP to prevent simultaneously selecting all the families selected by MP at the previous iteration. If the solution of the MP is feasible but split penalties occur, a *Penalty Cut* (Pc) is added to MP and an additional penalty term is added to its objective function stating that if the same set of families selected by the MP at the previous iteration is selected again, a correspondent split penalty has to be paid. The method iterates until the objective function of the optimal solution of MP is lower than the best feasible solution found by SP.

The third section of the table, indicated as CBC-MKr, reports the results attained by the new CBC approach described in Section 4.1. It contains three columns reporting the averages of the percentage %Os, the computation time T (s), and the number of Benders Cuts (indicated as C) added per instance, respectively. Ultimately, the bottom row of Table 2, for each column, provides the average calculated taking into account the different number of instances in the groups $G1$-$G4$.

The performance on these benchmark tests of both the CBC-SKr method and the baseline ILP Model has been studied by Mancini et al. (2021), showing that the Combinatorial Benders Cuts approach outperforms the ILP_MMdKFSP baseline both in terms of solution quality and execution time. Nevertheless, none of these two methods can optimally solve all instances in the assigned maximum computation time. The proposed CBC-MKr approach consistently performs better on all groups $G1$-$G4$. In fact, it optimally solves 100% of instances always using a much shorter computational time (i.e., on average CBC-MKr is about 470 times faster than the ILP solver and about 300 times compared to CBC-SKr), thus resulting far superior to the other two methods for all the group of instances included in the MCM2019 benchmark set. The nonparametric Kruskal–Wallis H-test (Kruskal and Wallis, 1952) followed by a *post hoc* analysis based on the Nemenyi test (Nemenyi, 1962) demonstrated CBC-MKr's clear superiority over competing approaches in terms of execution time. We observe that this result seems mainly due to the particular quality of the relaxation used, since the Benders cuts are added only in the case of group $G2$ containing large size higher dimensional instances, which consequently require a relatively longer calculation time. Besides this case, a correlation analysis suggests that, while the first two methods show an increasing computational effort passing from $G1$ to $G4$, the CBC-MKr performance seems to be only marginally affected by the different typology of instances showing a more flexible and robust behavior also due to the adoption of stronger cuts.

Tables 3–6 summarized the results obtained for the test instances of groups from $G5_1$ to $G5_4$, respectively. Each table is organized in three sub-tables dedicated to the detailed results of the three considered methods indicated as ILP_MMdKFSP, CBC-SKr, and CBC-MKr. In this case, the reported results refer (by rows) to each instance indicated in the first column (*Inst (m)*) with the indication of the considered number of families $m$. The value OF of the objective function obtained for each instance is reported for all the methods, and the required computation time T (s) in seconds. The OF value is reported in bold when the method certifies its optimality. The sub-table devoted to CBC-SKr reports the number of Feasibility and Penalty Cuts, indicated as Fc and Pc, occurred for each instance, whereas the CBC-MKr section shows the number C of Benders Cuts introduced by the novel method. Finally, the last row of each table reports the average results for the group of instances to which it refers.

Looking at the results of the $G5_1$ group of instances shown in Table 3, we note that for this case characterized by the lowest number of families (i.e., the set presents an average value of $m = 5.4$), all the methods were able to solve all the instances optimally. Nevertheless, it can be seen that CBC-MKr is much faster than the other two approaches for all instances. On average, the new method requires a computation time equal to approximately 9% of that used by the solver on the ILP model and approximately 3.4% of that required by CBC-SKr. In this group of test instances, we find that CBC-SKr has a relatively worse behavior among the three compared methods. More specifically, it does not generate Feasibility Cuts but always introduces Penalty Cuts with an average of 4.9 additional cuts per instance. On the contrary, CBC-MKr does not produce Benders Cuts showing that the MKr relaxation for this group of instances offers solutions that turn out to be optimal for MMdKFSP. Furthermore, a detailed analysis of the results shows that the families selected in the optimal solution are on average 3.4, but in no case are all the families selected.

Moving on to the results relating to the $G5_2$ group shown in Table 4, it can be seen that CBC-SKr and the baseline ILP_MMdKFSP solve to optimality only 60% of the instances within the time limit of 3600 seconds, whereas CBC-MKr rapidly solves all the cases taking an average of

2.64 seconds. In this case, the performances of ILP_MMdKFSP are on average the worst among the compared methods. CBC-SKr produces on average 44.60 Feasibility Cuts and 226.60 Penalty Cuts, while CBC-MKr introduces Benders Cuts only in 20% of cases that consequently require an higher computation time with respect to the previous group of instances. All other instances' features being equal, the increase in the number of families leads to an increase in the computational effort required, but this is particularly accentuated for the ILP model and CBC-SKr, while it remains much more contained for CBC-MKr. In this case, the average number of families per instance is 12.5, while the families selected in the optimal solution are on average 6.4, and they are never all selected.

This behavior is confirmed in the group of instances $G5_3$ where CBC-SKr performs better than ILP_MMdKFSP. However, even in this case CBC-MKr is still the best method by optimally and fastly closing all instances as in the group of cases $G5_2$. The computation time taken by CBC-MKr is again orders of magnitude lower than that required by the other two compared methods. Furthermore, the new method generates Benders Cuts in 40% of cases, highlighting that the algorithmic scheme is able to rapidly solve MMdKFSP joining the quality of the relaxation solutions and the effectiveness of the adopted cuts. In group $G5_3$, the instances have an average number of families $m = 12.7$, whereas the families selected in the optimal solution are 7.9 on average, and never selected altogether.

The superior performance of CBC-MKr is even more marked in the case of the $G5_4$ group of instances (generated with an expected number of families $m = 20$) whose results are shown in Table 6. In fact, it solves all instances quickly and without generating cuts, whereas CBC-SKr is unable to certify the optimality for any instance within the time limit, and the ILP model optimally solves only 50% of instances using on average more than 2000 seconds compared to 1.75 seconds required by CBC-MKr. A detailed analysis of the results shows that the families selected in the optimal solution are on average 10.80, while the average number of families in the instances is $m = 20.1$. As in the previous cases, in the optimal solutions the families are never all selected.

A nonparametric statistical analysis of the results, similar to that presented for the G1-G4 groups and extended to the G5 group instances, demonstrates the superiority of the CBC-MKr method in terms of computation time. The boxplot in Fig. 1 graphically shows the three methods' calculation times for all instances (the dashed purple line identifies the average values). Note that the figure features a *symlog* scale abscissae axis so that time values lower than one second can be better distinguished.

The results obtained in the experimental campaign show that the novel approach CBC-MKr is the only method able to solve all the considered test instances optimally. In addition, CBC-MKr has proven to be a very fast algorithm due to a specific relaxation that is particularly tight, and to the effectiveness of the considered Benders Cuts. The computational time required by CBC-MKr is always a small fraction of that required by the other two compared methods. These results are confirmed in the new set of instances $G5$, where in cases with very low or very high $m$ values CBC-MKr shows a better behavior regarding the computation time, due to the particularly good solutions obtained by relaxation MKr. Although in the instances characterized by intermediate $m$ values in about 30% of cases, some Benders Cuts (generally few) are introduced by the CBC-MKr algorithm before reaching the optimal solution, thus requiring a relatively higher computational time. It can be observed that extreme values of $m$ lead the instances to seem those of other problems of the knapsack family or even of assignment or packing problems.

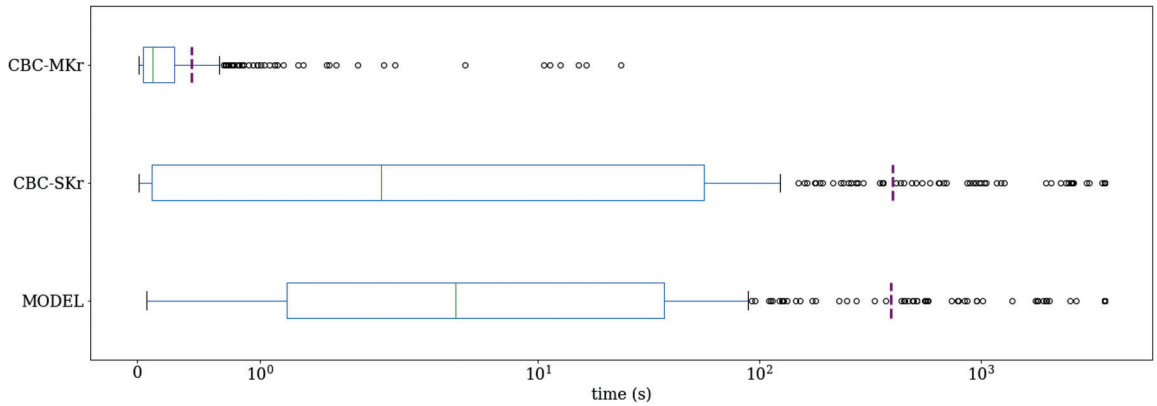


Fig. 1. Boxplot of computation time distributions for the experimental campaign. The purple line represents the mean value for each method.

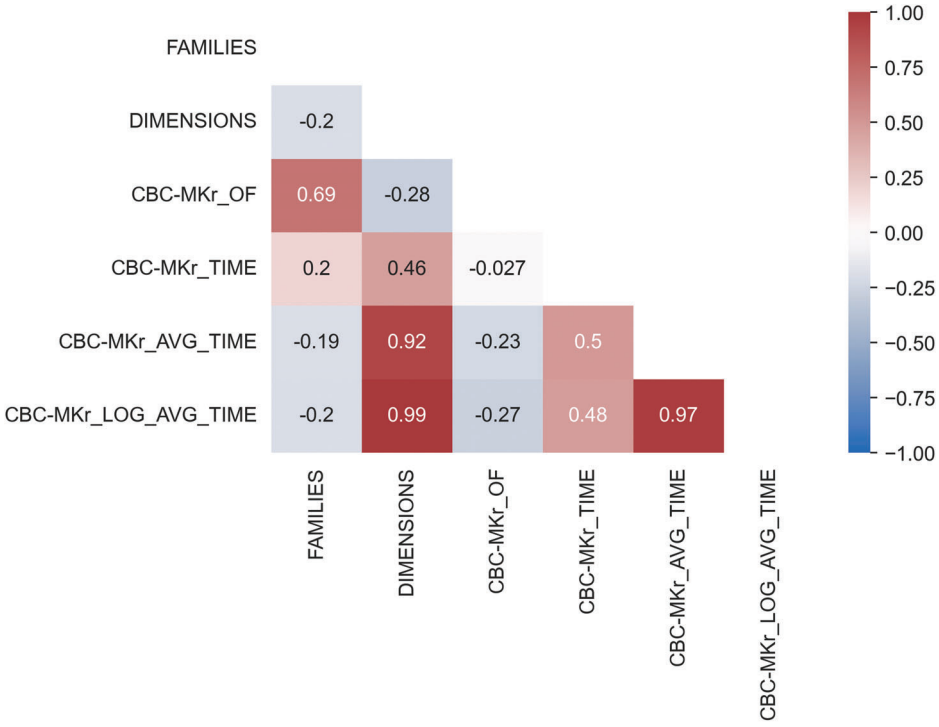


Fig. 2. Correlation matrix for selected variables from results of G2 instances.

Finally, the dependence of the number of dimensions ($D$) on CBC-MKr's execution times is analyzed. In this regard, Fig. 2 presents the correlation matrix computed from the results of the experiments on the G2 group of instances ($n = 600, |\mathcal{K}| = 10$). It is noticeable that while the objective function (CBC-MKr_OF) is positively correlated to the number of families $m$ (FAMILIES), the number of dimensions $D$ (DIMENSIONS) affects the execution times of the algorithm

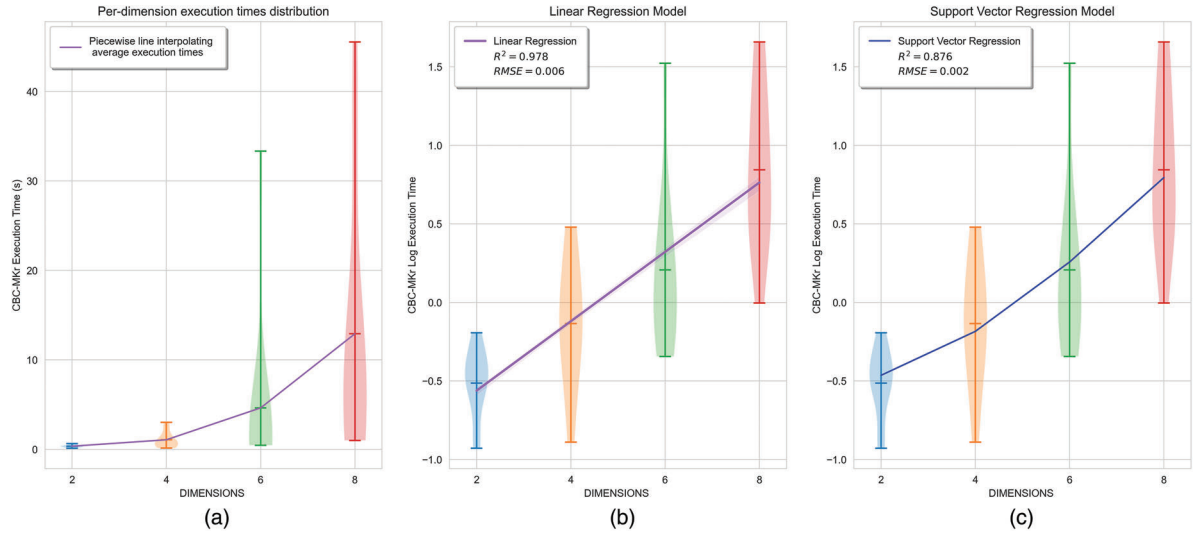


Fig. 3. (a) Violin plot of CBC-MKr's per-dimension execution times distributions. A nonlinear increasing trend is noticeable. (b) Line plot (linear regression) superimposed on a violin plot of CBC-MKr's per-dimension log execution times distributions. (c) Line plot (SVR) superimposed on a violin plot of CBC-MKr's per-dimension log execution times distributions.

(CBC-MKr_TIME) with a correlation coefficient equal to 0.46. Although at first glance correlation seems to be somewhat limited, it must be considered that the covariate DIMENSIONS is categorical ($D \in \{2, 4, 6, 8\}$) and multiple data points are associated to the same value of $D$. Therefore, it is sensible to consider the average execution time to identify a more explicit dependence. And indeed, the correlation coefficient between DIMENSIONS and CBC-MKr_AVG_TIME is conspicuously higher, that is 0.92. Furthermore, the per-dimension time distributions plot (see Fig. 3a) suggests a nonlinear, quasi-exponential increasing trend of time averages versus the number of dimensions. We then applied a base 10 logarithmic transformation to the time values and calculated the average of the data points thus obtained. The transformation is meant to reduce nonlinearity; notably, the correlation coefficient between DIMENSIONS and CBC-MKr_LOG_AVG_TIME rises to 0.99.

To complete discussion of the computational results, a linear regression model (CBC-MKr_LOG_AVG_TIME $\approx$ DIMENSIONS) was fit and compared with a support vector regression model (SVR). As expected, see Fig. 3b, the DIMENSIONS covariate is statistically significant ($p < 0.001$) and the linear model features a very high $R^2$, that is, 0.978 and a very low Root-Mean-Square Error (RSME), that is, 0.006. Although these results explain more than satisfactorily the dependence of $D$ on the CBC-MKr's average execution times, a final comparison is performed using a Support Vector Regression (SVR) model with Radial Basis Function (RBF) Kernel to assess the impact of the remaining nonlinearity on the studied dependence: as it can be seen from Fig. 3b–c, although the SVR provides a lower error (RSME = 0.002) the difference from the linear model is only marginal.

We complete the analysis of the behavior of the proposed algorithm with a final experiment, the results of which are given below, aimed at assessing the limits of applicability of the CBC-MKr algorithm. To this end, we use a new dataset, labeled $G6$, containing instances crafted

Table 7
Computational results of CBC-MKr on $G6_{10}$ and $G6_{20}$ instances. For all instances $n = 100$

| | OF | T(s) | C | F |
|---|---|---|---|---|
| $G6_{10}$ | 359.1 | 31.49 | 45.2 | 5.3 |
| $G6_{20}$ | 646.3 | 308.39 | 46.3 | 10.9 |

specifically to challenge the algorithm. More in detail, *G*6 comprises two groups of 10 instances each, referred to as $G6_{10}$, featuring 10 knapsacks, and $G6_{20}$ in which 20 knapsacks are used. In both cases, bi-dimensional knapsacks/items are considered. Each instance is characterized by a set of 100 items organized in 33 families with at most four items each. For each item, both sizes are independently sampled from $U[200, 300]$, whereas the knapsacks' sizes are drawn from $U[400, 600]$. In this setting, the per-dimension sizes of families and knapsacks are comparable. It is to be expected that such a *coarse granularity* makes packing the items more challenging as the relaxed solution (where a family is treated as a single, yet divisible, super-item) might be infeasible. As a consequence, CBC-MKr is expected to generate on these instances a larger number of cuts with respect to instances with *fine granularity* (i.e., $C_k^d \gg c_i^d \quad \forall k \in \mathcal{K}, i \in \mathcal{I}, d \in D$). Instances of this type, with coarse granularity, are realistic as they may arise in scenarios pertaining to the allocation of service-based applications in an edge computing environment. Indeed, in such circumstances, it is likely that nodes (knapsacks) have limited resources, comparable to those required to run a minimal set of applications.

Table 7 summarizes the results obtained on the set *G*6 reporting for $G6_{10}$ and $G6_{20}$, separately, the averages of the objective function (OF), computation time in seconds (T (s)), number of generated cuts (C), and the number of selected families (F). Examining the reported data, we note that on set $G6_{10}$, CBC-MKr requires on average 45 iterations to converge to the optimum. Average computational time is around 30 seconds, that is, at least one order of magnitude larger than the time required to address instances with a similar number of items and knapsacks in previous experiments (e.g., *G*5). Moreover, when the number of knapsacks doubles ($G6_{20}$), the number of families to be selected doubles accordingly, extending the space of feasible solutions and making the problem even more challenging. Indeed, even if the number of cuts required to converge is roughly unchanged for the two considered datasets (since the instances' granularity is the same in $G6_{10}$ and $G6_{20}$), when applied on instances of $G6_{20}$ CBC-MKr takes about seven seconds to complete a single iteration, which is $10x$ the time required for the case with 10 knapsacks. Such rapid growth in computation time (as a function of the number of knapsacks) indicates that it is possible to generate realistic instances that would be prohibitive even for the CBC-MKr algorithm and call for other solution methods (e.g., heuristics).

To conclude, the analysis of these results shows that the CBC-MKr algorithm, in addition to offering excellent performances, is particularly flexible and robust with respect to the parameters and configurations of the instances. The success of this method is due to a very tight relaxation used in MP, as well as to the adoption of very effective cuts able to simultaneously exclude an exponential number of solutions from the solution space.

On the basis of the results obtained, the CBC-MKr proves to be an excellent method to deal with the MMdKFSP. Moreover, it appears very promising to face also different but similarly structured

problems and to play a relevant role in the design of more complex algorithmic schemes. An example is given in the following Section 5.2.1, where the extension to the Fixed-Charge MMdKFSP is considered.

### 5.2.1. Application of CBC-MKr to the fixed-charge MMdKFSP

A relevant feature of the proposed CBC-MKr algorithm is represented by its possible application to the more general Fixed-Charge version of MMdKFSP (indicated as FC-MMdKFSP) as an extension of the original problem in which a specific fixed cost must be paid for each knapsack used in the solution. Therefore, the problem is also to decide the set of knapsacks to use and to assign items to them, maximizing the total net profit given by item profits minus knapsack costs and family-split penalties.

Introducing a set of binary variables ($w_k$) associated with the utilization of each knapsack $k \in \mathcal{K}$, and a corresponding fixed cost $\phi_k$, the mathematical formulation of the MMdKFSP variant can be written as follows.

$$\text{(ILP\_FC-MMdKFSP)} \qquad \max \sum_{j|\mathcal{F}_j \in \mathcal{F}} p_j x_j - \sum_{j|\mathcal{F}_j \in \mathcal{F}} \delta_j u_j - \sum_{k \in \mathcal{K}} \phi_k w_k, \tag{30}$$

Subject to:

$$\sum_{k \in \mathcal{K}} y_{ik} = x_j \quad \forall i \in \mathcal{I}, \quad \forall j | i \in \mathcal{F}_j, \tag{31}$$

$$\sum_{i \in I} c_i^d \, y_{ik} \leq C_k^d \quad \forall k \in \mathcal{K}, \quad \forall d \in \{1, \ldots, D\}, \tag{32}$$

$$z_{jk} \geq \frac{1}{|\mathcal{F}_j|} \sum_{i \in \mathcal{F}_j} y_{ik} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}, \tag{33}$$

$$u_j \geq \frac{1}{|\mathcal{K}| - 1} \left( \sum_{k \in \mathcal{K}} z_{jk} - 1 \right) \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \tag{34}$$

$$w_k \geq z_{jk} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, , \quad \forall k \in \mathcal{K} \tag{35}$$

$$x_j, u_j \in \{0, 1\} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \tag{36}$$

$$z_{jk} \in \{0, 1\} \quad \forall j | \mathcal{F}_j \in \mathcal{F}, \quad \forall k \in \mathcal{K}, \tag{37}$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in \mathcal{I}, \quad \forall k \in \mathcal{K} \tag{38}$$

$$w_k \in \{0, 1\} \quad \forall k \in \mathcal{K}. \tag{39}$$

Our CBC-MKr method can be easily adapted to address FC-MMdKFSP, by substituting the original objective function (16) with (30). Conversely, CBC-SKr cannot be adapted to deal with

Table 8
Comparison of computational times (in seconds) required by ILP and CBC-MKr on the FC-MMdKFSP instances

| | ILP | CBC-MKr |
|---|---|---|
| $G5_{4A}$ | 2021.62 | 1.02 |
| $G5_{4B}$ | 2012.12 | 1.38 |
| $G5_{4C}$ | 1999.89 | 1.00 |
| $G5_{4D}$ | 2022.01 | 1.19 |

this extension. In fact, in CBC-SKr, the MP considers a single meta-knapsack representing the union of all the knapsack, therefore it is not possible to quantify the fixed charged costs associated to a specific solution of the MP. In order to assess the performance of CBC-MKr on FC-MMdKFSP, we generated four additional sets of instances, starting from set $G5_4$ and adding different types of fixed charges (FCs). Set $G5_{4A}$ considers high FCs, uniformly distributed in the interval [40,70], $G5_{4B}$ adopts high homogeneous FC equal to 50, $G5_{4C}$ low FCs, uniformly distributed in the interval [10,40], whereas $G5_{4D}$ includes low homogeneous FC set equal to 25. In Table 8, we report a comparison between the ILP solution using Xpress 7.9 and CBC-MKr. Results show a clear dominance of CBC-MKr that is able to solve all the instances in around 1 second, while ILP require around 2000 seconds. The type of FCs considered (homogeneous/heterogeneous and low/high) does not have a significant impact on the computational times required by both approaches. We can conclude that CBC-MKr is a very effective method for FC-MMdKFSP too. Moreover, the optimal solution of MP turned out to be optimal for the original problem in all the tested instances, which prove the tightness of the proposed relaxation also on FC-MMdKFSP.

## 6. Conclusions

This work deals with the MMdKFSP, a problem considered in the literature as a variant of the more classic Multi-Knapsack and problems Multidimensional Knapsack problems), characterized by the fact that items are organized into families. Families impose a selection constraint on items, that is, either all family items are selected or none is. Also, the problem requires that if a family is selected, it is more convenient, from a profit point of view, to assign its items to the same knapsack; otherwise, it will incur a penalty.

This problem is receiving a growing interest from different application fields, which calls for adequate attention from the modeling and algorithmic point of view.

In this paper, we propose a novel exact and fast approach to solve the problem. This approach is based on a dedicated problem relaxation (MKr) and on a specific scheme of the Combinatorial Benders Cuts method (named CBC-MKr). A number of sets of test instances with different sizes and layouts (either available from the literature or developed in this research work) are used as a benchmark to compare the performance of the proposed algorithm with those of both the state-of-the-art MMdKFSP algorithm, and a cutting edge solver processing the baseline ILP formulation. We have compared the algorithms according to criteria of efficiency, optimality of results,

and scalability, as is evident from the analysis reported in Section 5.2, CBC-MKr statistically and significantly outperforms its competitors.

The computational experiments show that the proposed specific relaxation procedure can provide particularly good upper bounds to be used in an algorithmic scheme that is substantially different from the state-of-the-art algorithm available in the literature (Mancini et al., 2021) and from the classic CBC method (Benders, 1962). The proposed CBC algorithm offers excellent computational performances and is particularly flexible and robust with respect to the parameters and configurations of the instances, including the possibility of solving the fixed-charge variant of MMdKFSP.

The results obtained in this paper suggest that the proposed algorithmic approach could be extended or generalized to tackle structurally related problems, including different assignment, packing and knapsack problems arising in various application contexts such as manufacturing, transportation, logistics, and services.

Possible future research directions may involve the design, implementation, and testing of new algorithms, either exact or heuristic, to efficiently address challenging instances representing special cases or that are, for example, orders of magnitude larger than those considered in this work. These extensions could include the study of stronger combinatorial Benders cuts.

## Acknowledgments

Open Access Funding provided by Università degli Studi di Roma La Sapienza within the CRUI-CARE Agreement.

## References

Adany, R., Feldman, M., Haramaty, E., Khandekar, R., Schieber, B., Schwartz, R., Shachnai, H., Tamir, T., 2013. All-or-nothing generalized assignment with application to scheduling advertising campaigns. In Goemans, M., Correa, J. (eds) *Integer Programming and Combinatorial Optimization*. Springer, Berlin, pp. 13–24.

Amiri, A., 2020. A Lagrangean based solution algorithm for the knapsack problem with setups. *Expert Systems with Applications* 143, 113077.

Amiri, A., Barkhi, R., 2021. A Lagrangean based solution algorithm for the multiple knapsack problem with setups. *Computers & Industrial Engineering* 153, 107089.

Benders, J., 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4, 238–252.

Bruglieri, M., Mancini, S., Peruzzini, R., Pisacane, O., 2021. The multi-period multi-trip container drayage problem with release and due dates. *Computers & Operations Research* 125, 105102.

Cacchiani, V., Iori, M., Locatelli, A., Martello, S., 2022a. Knapsack problems - an overview of recent advances. part i: single knapsack problems. *Computers & Operations Research* 143, 105692.

Cacchiani, V., Iori, M., Locatelli, A., Martello, S., 2022b. Knapsack problems - an overview of recent advances. part ii: multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research* 143, 105693.

Ceselli, A., Righini, G., 2006. An optimization algorithm for a penalized knapsack problem. *Operations Research Letters* 34, 4, 394–404.

Chekuri, C., Khanna, S., 2000. A ptas for the multiple knapsack problem. In Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, pp. 213–222.

Chen, L., Zhang, G., 2018. Packing groups of items into multiple knapsacks. *ACM Transactions of Algorithms* 14, 4, 51:1–51:24.

Chen, Y., Hao, J.K., 2014. A "reduce and solve" approach for the multiple-choice multidimensional knapsack problem. *European Journal of Operational Research* 239, 2, 313–322.

Chu, Y., Xia, Q., 2004. Generating benders cuts for a general class of integer programming problems. In Régin, J.-C., Rueher, M. (eds) *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, Berlin, pp. 127–141.

Codato, G., Fischetti, M., 2006. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research* 54, 756–766.

Della Croce, F., Pferschy, U., Scatamacchia, R., 2019. New exact approaches and approximation results for the penalized knapsack problem. *Discrete Applied Mathematics* 253, 122–135.

Della Croce, F., Salassa, F., Scatamacchia, R., 2017. An exact approach for the 0-1 knapsack problem with setups. *Computers & Operations Research* 80, 61–67.

Dell'Amico, M., Delorme, M., Iori, M., Martello, S., 2019. Mathematical models and decomposition methods for the multiple knapsack problem. *European Journal of Operational Research* 274, 3, 886–899.

Ferreira, C., Martin, A., Weismantel, R., 1996. Solving multiple knapsack problems by cutting planes. *SIAM Journal on Optimization* 6, 858–877.

Fréville, A., 2004. The multidimensional 0-1 knapsack problem: an overview. *European Journal of Operational Research* 155, 1, 1–21.

Furini, F., Monaci, M., Traversi, E., 2018. Exact approaches for the knapsack problem with setups. *Computers and Operations Research* 90, 208–220.

Han, B., Leblet, J., Simon, G., 2010. Hard multidimensional multiple choice knapsack problems, an empirical study. *Computers & Operations Research* 37, 1, 172–181.

Hifi, M., Michrafy, M., Sbihi, A., 2004. Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *Journal of the Operational Research Society* 55, 12, 1323–1332.

Hooker, J., Ottoson, G., 2003. Logic-based benders decomposition. *Mathematical Programming* 96, 33–60.

Kataoka, S., Yamada, T., 2014. Upper and lower bounding procedures for the multiple knapsack assignment problem. *European Journal of Operational Research* 237, 2, 440–447.

Kellerer, H., Pferschy, U., Pisinger, D., 2004. *Knapsack Problems*. Springer, Berlin.

Kruskal, W., Wallis, W., 1952. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association* 47, 260, 583–621.

Lust, T., Teghem, J., 2012. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research* 19, 4, 495–520.

Mancini, S., Ciavotta, M., Meloni, C., 2021. The multiple multidimensional knapsack with family-split penalties. *European Journal of Operational Research* 289, 3, 987–998.

Mancini, S., Gansterer, M., 2021. Vehicle scheduling for rental-with-driver services. *Transportation Research Part E: Logistic and Transportation Review* 156, 102530.

Mansi, R., Alves, C., Valerio de Carvalho, J., Hanafi, S., 2013. A hybrid heuristic for the multiple choice multidimensional knapsack problem. *Engineering Optimization* 45, 8, 983–1004.

Mansini, R., Zanotti, R., 2020. A core-based exact algorithm for the multidimensional multiple choice knapsack problem. *INFORMS Journal on Computing* 32, 4, 855–1186.

Martello, S., Monaci, M., 2020. Algorithmic approaches to the multiple knapsack assignment problem. *Omega* 90, 102004.

Martello, S., Toth, P., 1990. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Hoboken, NJ.

McLay, L., Jacobson, S., 2007. Knapsack problems with setups. *Computational Optimization and Applications* 37, 1, 35–47.

Michel, S., Perrot, N., Vanderbeck, F., 2009. Knapsack problems with setups. *European Journal of Operational Research* 196, 3, 909–918.

Nauss, M., 1978. The 0-1 knapsack problem with multiple-choice constraints. *European Journal of Operational Research* 2, 125–131.

Nemenyi, P., 1962. Distribution-free multiple comparisons. *Biometrics* 18, 2, 263.

Pferschy, U., Scatamacchia, R., 2018. Improved dynamic programming and approximation results for the knapsack problem with setups. *International Transactions in Operational Research* 25, 667–682.

Pisinger, D., 1999. An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research* 114, 528–541.

Shachnai, H., Tamir, T., 2001. Polynomial time approximation schemes for class-constrained packing problems. *Journal of Scheduling* 4, 313–338.

Shojaei, H., Basten, T., Geilen, M., Davoodi, A., 2013. A fast and scalable multidimensional multiple-choice knapsack heuristic. *ACM Transactions on Design Automation of Electronic Systems* 18, 4, 51:1–51:24.

Sun, X., Ansari, N., Wang, R., 2016. Optimizing resource utilization of a data center. *IEEE Communications Surveys & Tutorials* 18, 4, 2822–2846.

Yamada, T., Takeoka, T., 2009. An exact algorithm for the fixed-charge multiple knapsack problem. *European Journal of Operational Research* 192, 700–705.