Effect of the local branching strategy on the descent method: The case of the generalized multiple knapsack with setup

Samah Boukhari, Isma Dahmani, Mhand Hifi

Version of Record: https://www.sciencedirect.com/science/article/pii/S0360835222000043
Manuscript_c030183d1366d7458b0f9ef7d991635d

Effect of the Local branching Strategy on the Descent Method: The Case of the Generalized Multiple Knapsack with Setup

Samah Boukhari[†], Isma Dahmani[†] and Mhand Hifi[‡,⋆]

All authors are listed in alphabetical order of their last name, following the standard order used by the Operational Research community in France.

[†]**Samah Boukhari**

LaROMaD, USTHB BP 32 El Alia 16111 Bab Ezzouar, Algiers, Algeria

E-mail: boukhari.samah.ro@gmail.com

[†]**Isma Dahmani**

LaROMaD, USTHB BP 32 El Alia 16111 Bab Ezzouar, Algiers, Algeria

E-mail: dahmani.isma@gmail.com

[‡]**Mhand Hifi**

[⋆]Corresponding author.

EPROAD EA 4669, UPJV, 7 rue du Moulin Neuf, 80000 Amiens, France

E-mail: hifi@u-picardie.fr

Declarations of interest.

The authors do not have any possible conflicts of interest.

Abstract

In this paper, we investigate the use of the local branching strategy into an iterative descent method for tackling the generalized multiple knapsack problem with setup. An instance of the problem is composed of a set of classes containing items, and a set of available knapsacks. Its objective consists in selecting the subsets of items belonging to the classes with a maximum objective value: each item is characterized with its profit and weight while a class is characterized with its cost such that a given item may be selected whenever its corresponding class is activated, and an item can be configured (setup) for a single knapsack. The proposed iterative method starts by solving a series of reduced subproblems built by adding a series of cardinality constraints. Next, the local branching strategy is iteratively employed for highlighting the efficiency of the method. Finally, the behavior of the method is evaluated on a set of instances extracted from the literature, where its achieved bounds are compared to those reached by the best methods available in the literature. A statistical analysis is also provided showing the high performance of the method. New bounds are discovered.

Keywords: Branching; Integer Programming; Knapsack; Learning; Setups.

1. Introduction

The knapsack problem occurs in several real-world situations, like cutting and packing (Chen et al., 2019; Liu et al., 2021), cryptography (Merkle & Hellman, 1978), logistics (Perboli et al., 2014), and others (Plata-González et al., 2019). Such a problem plays, on the one hand, a central role in modeling more higher NP-hard combinatorial optimization problems, where the designed models may be used as a guiding strategy for designing powerful exact and approximate methods. On the other hand, tackling large-scale instances is often important, especially when considering real-world application for which both runtime and quality of reached solutions are crucial. In recent years, simple deterministic and

stochastic heuristics were also used for solving several knapsack problems while it has been observed that the hybridization of solution procedures may be considered as a very promising research issue. Nevertheless, the difficulty which may be encountered when applying such methods may be related to the runtime consumed.

In this paper, we focus on solving the Generalized Multiple Knapsack Problem with Setup (GMKPS), which belongs to the NP-hard knapsack family. On the one hand, GMKPS is a generalization of multiple knapsack problem (MKP), where items belong to disjoint classes and can be processed in multiple knapsacks. The activation of a class induces setup costs and resource consumptions (setup time), which has a negative impact on both the capacity constraint and the objective function. On the other hand, it can be viewed as an extended version of the single Knapsack Problem with Setup (KPS) such that multiple knapsacks with setups are considered.

An instance of GMKPS is characterized by a set of T knapsacks (such that $t \in \{1, \ldots, T\}$) and a set of N disjoint classes (families) of items. A class $i \in \{1, \ldots, N\}$ is characterized by n_i items and a knapsack-dependent integer setup cost f_{it}, and an integer s_i denoting its capacity consumption. Each item j, $j \in \{1, \ldots, n_i\}$, of a family i is associated with a knapsack-dependent profit p_{ijt} and a capacity consumption w_{ij}. Further, an item can be selected only if its corresponding class is activated, and a n item can only be setup into one knapsack. In this problem the activation of a class incurs a knapsack-dependent setup cost, which should be considered in both the objective function and constraints. The objective of the problem consists in selecting appropriate items, from different disjoint classes, to belong to the knapsack with a maximum objective value, and without violating the capacity constraints.

Let x_{ijt} equal to 1 if item j of family(class) i is placed in period t, 0 otherwise. Setup binary variables y_{it} equal to 1 if family i of knapsack t is activated, 0 otherwise. Thus, the formal description of GMKPS (noted P_{GMKPS}) follows:

$$\mathrm{P}_{GMKPS}: \quad \max \quad \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{n_i} p_{ijt}x_{ijt} + \sum_{t=1}^{T}\sum_{i=1}^{N} f_{it}y_{it} \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\Big(\sum_{j=1}^{n_i} w_{ij}x_{ijt} + s_i y_{it}\Big) \leq R_t, \ \forall t \in \mathcal{T} \tag{2}$$

$$x_{ijt} \leq y_{it}, \quad \forall i \in \mathcal{N}, \ \forall j \in \{1, ..., n_i\}, \ \forall t \in \mathcal{T} \tag{3}$$

$$\sum_{t=1}^{T} x_{ijt} \leq 1, \quad \forall i \in \mathcal{N}, \ \forall j \in \{1, ...n_i\} \tag{4}$$

$$x_{ijt} \in \{0,1\}, \ y_{it} \in \{0,1\}, \ \forall i \in \mathcal{N}, \ \forall j \in \{1, ..., n_i\}, \ \forall t \in \mathcal{T}. \tag{5}$$

The objective function (Equation 1) maximizes the profit of selected items excluding the fixed setup costs of selected families (classes). Constraints ((2)) guarantee that the weight of selected items in each knapsack, augmented with their setup capacity consumption, does not exceed the knapsack capacity R_t. Constraints (3) ensure that an item j is selected in the knapsack only if its family i of period t is activated. Constraints (4) indicates that any item j of a class i is setup at most in one knapsack. Finally, constraints (5) represent the integrality of the decision variables.

The remainder of the paper is organized as follows. The background is described in Section 2. The designed approach which combines both descent method and local branching is presented in Section 3. Section 3.1 discusses the variable neighborhood descent method. A restricted linear relaxation of the original problem is discussed in Section 3.2, where a starting solution is provided by applying a tailored procedure. Section 3.3 summarizes the principle of the basic local branching. Section 3.4 exposes a new process that is introduced for reaching a series of feasible solutions; that is based upon a series of constraints used as local branches into an iterative descent method, like variable neighborhood descent.. Finally, Section 4 exposes the experimentations conducted on three sets of benchmark instances taken from the literature. Its behavior is also analyzed in the same section, where its provided results are compared to those achieved by the best methods available in the literature and the state-of-the-art Cplex solver.

A new hybrid method is designed for approximately solving GMKPS. The main principle of the method is based upon injecting a series of local branching into an iterative descent method, where a reference solution is built throughout a special mixed integer programming. The proposed method employs the following features:

1. To build a *reference solution* used as a starting solution for the designed method. This solution is obtained by solving a starting mixed integer programming which is combined with a collecting procedure.
2. Then, the following two stages are used:
   (a) To build a relaxed problem and to solve it by using a local branching-based algorithm and,
   (b) To solve a new reduced problem by injecting information collected from the resolution of the mixed integer model solved at Step 2a.
3. Steps 2a and 2b are iterated till matching the final stopping condition.

2. Related work

GMKPS belongs to the knapsack problem family, which is one of the old problems belonging to the combinatorial optimization problems. Such a problem can model several real-world situations, where its formalism fits well with the most complex problems while the academic studies considered practical situations as references. Because of the NP-hardness of the majority of problems belonging to the knapsack problem family, any exact method may be used for tackling some small and medium sized instances and so, the availability of effective heuristics and meta-heuristics are of paramount importance.

GMKPS is a more complex version of the well-known knapsack problem, where despite its NP-hardness, to the best of our acknowledgement, there are few papers tackling this problem in the literature. Among these papers, we cite Adouani et al. (2020) who designed an efficient heuristic which combines variable neighborhood descent and integer linear programming. The local search-based procedure was applied for assigning classes to knapsacks while the integer programming-based procedure was used for selecting the items in each knapsack. Finally, their method was evaluated on benchmark instances of the literature, where its provided results were compared to those achieved by the state-of-the-art Cplex solver.

Adouani et al. (2019) designed a variable neighborhood search for solving the multiple choice knapsack problem with setup, another version of the the knapsack with setups. The method is based upon hybridization of stochastic local search with solving a series of small-sized subproblems with a tailored solver. On the one hand, the stochastic local search applies the so-called perturbation strategy where some items are randomly removed from the current solution. On the other hand, an induced reduced subproblem is then solved with the Cplex solver for completing the current partial solution. Both strategies were embedded into an iterative search till matching a final predefined stopping criteria. Finally, their method was experimentally analyzed on 120 instances, and their provided results were compared to those achieved by the state-of-the-art Cplex solver.

Other studies have tackled several versions of the problems belonging to the knapsack family, where setups are considered as constraints. Among these problems, we cite a Lagrangean relaxation-based heuristic that has been designed in Amiri (2020), where the method was tailored for large-scale instances of the knapsack problem with setup. His method follows the standard adaptation of the Lagrangean relaxation, where a series of local optima are localized with a descent method and converted into feasible solutions. The

behavior of the method was evaluated on both standard set of benchmark instances and huge-scale randomly generated instances (containing at most 500 classes and two million of items); its achieved results were compared to those achieved by the best available method in the literature and the state-of–the-art Cplex solver. For the same problem, Boukhari et al. (2020) proposed a tailored local branching-based heuristic, where the method hybridizes both mixed linear relaxation and local branching. The mixed linear relaxation was solved by calling a special black-box solver while the local branching tried to intensify each solution at hand by adding a series of local-branching constraints. The performance of that method was evaluated on benchmark instances of the literature and new large-scale ones; its provided results were compared to those provide by the Cplex solver and the best available methods.

Chebil & Khemakhem (2015) studied the problem related to the knapsack problem with setup. In their work, a straightforward of the classical dynamic programming procedure was proposed for exactly solving that problem, where the algorithm performs in a pseudo-polynomial time complexity. A tailored converting formulation was also considered in order to decrease the size of the storage capacity, which is often expensive when using such type of approach. For the same problem, Khemakhem & Chebil (2016) designed a special truncated tree-search for approximately solving it. The method applies an avoid duplication technic which consists in reformulating the original problem into a particular integer programming. The experimental part showed the effectiveness of the proposed method, especially its effect when using the avoiding duplication technic.

Furini et al. (2018) solved the continuous relaxation of that problem, where linear-time algorithms were proposed for optimally solving it, and different integer linear programming formulations were considered. As mentioned in their experimentations, their algorithms outperform the dynamic programming method and the state-of-the-art Cplex solver.

Della Croce et al. (2017) designed an optimal method that handles the structure of the original model of the knapsack problem with setup. The search procedure employed a partitioning strategy, where the decision variables were split into two levels. Thus, a fixation strategy has been added in order to reduce the subproblem at hand while the blackbox solver is called for solving the resulting subproblem. In their experimental part, the authors pointed the competitiveness of their method, especially when comparing its provided results to those reached by both the blackbox solver and the dynamic programming-based approach.

Boukhar et al. (2020) studied the effect of the cardinality constraint when adding it to an iterative local branching-based method. According to the used linear mixed integer model, a series of branches have been introduced to speed up the search process. It was noticed that such constraints acted in an interesting way when some decision variables were targeted. In their experimental part, the authors showed the interest of such a strategy, especially on benchmark instances of the literature.

Yang (2006) studied the problem with three knapsack constraints, where branch-and-bound procedure was proposed. Their method applies the well-known greedy procedure for providing a starting solution (lower bound) of the search process while a linear relaxation formulation was used for bounding the search with an upper bound for solving the multiple-knapsack with setups. The experimental part showed that their method had a good behavior, especially several problem instances.

In Lahyan et al. (2019) the authors designed a multilevel matheuristic for tackling large-scale instances related to the multiple knapsack problem with setup problem. The method is based on two stages: (i) reducing the original problem into a series of subproblems, where each class contains one item, (ii) the linear relaxation of the problems built were solved such that a current feasible solution is collected. In order to enhance the quality of the solution at hand, a tabu strategy was applied. In their experimental part, the authors pointed the competitiveness of their method when comparing the results provided by the method to those reached by the best methods available in the literature.

Finally, Boukhari et al. (2022) designed a hybrid algorithm that combines a solution resulted from solving a mixed integer linear relaxation and a series of single knapsack problems. Such a method was also reinforced by injecting a series of valid constraints for realizing a powerful method. In their experimental results, the authors studied the behavior of the method on benchmark instances of the literature. Its provided results were compared to those obtained by the best methods published in the literature and the state-of-the-art Cplex solver; several bounds have been reached.

3. Local branching as a learning strategy for GMKPS

3.1. A basic variable neighborhood descent

On the one hand, a Variable Neighborhood Search (VNS) (Brimberg et al., 2000) is an approximate descent method which has been extensively used for tackling complex problems arises in real-world applications and academic optimization problems. It is based

upon two strategies, where tailored neighborhoods are often employed: (i) a stochastic descent strategy used for enhancing the solution at hand until converging to a local optimum, and (ii) a stochastic perturbation strategy (shakings), which diversify the search process. On the other hand, the Variable Neighborhood Descent (VND) (Hansen & Mladenović, 2003) method may be viewed as a variant of VNS, where neighborhoods are replaced in a deterministic way. In this case, let $N_k,\ k = 1, \ldots, k_{max}$, be the successive neighborhoods to be explored by the method; thus, Algorithm 1 describes the main steps of the basic VND, where neighborhoods are called in a deterministic way.

Algorithm 1 Steps of the basic VND

Require: An instance of the problem with a starting solution $\underline{x}$ of objective value $z(\underline{x})$.
Ensure: A (near)optimal solution $x^\star$ with its objective value $z(x^\star)$.

```
Determine a series of neighborhood structures to use N_k, k = 1, ..., k_max.
Set k = 1, and x* = x
repeat
    Explore the current neighborhood N_k(x) and let x' the best neighbor around x.
    if x' is enhanced when compared to x's objective value z(x) then
        Update x with x', and restart the search with the initial neighborhood, i.e. k = 1
        Update x* if x' is better than x*.
    else
        increment the neighborhood index k
    end if
until (k = k_max)
return x*
```

3.2. A reference solution for the descent method

Because the Descent Method (DM), proposed in this work, needs a starting reference solution, we then discusses how that solution can be provided throughout solving a linear relaxation of P_{GMKPS}. We do it by calling the following two-phase procedure:

Phase 1. By applying modifications on the original model P_{GMKPS} according to the items j of the class i with respect to profit and weight or $p'_{it} = \sum_{j=1}^{n_i} p_{ijt}$ and $w'_{it} = \sum_{j=1}^{n_i} w_{ijt}$, $\forall i \in \mathcal{N},\ \forall t \in \mathcal{T}$, the problem P'_{GMKPS} may be rewritten as follows:

$$\mathrm{P}'_{GMKPS}: \quad \max \quad \sum_{t=1}^{T}\sum_{i=1}^{N} p'_{it}x_{it} + \sum_{t=1}^{T}\sum_{i=1}^{N} f_{it}y_{it} \tag{6}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} w'_i x_{it} + s_i y_{it} \leq R_t,\ \forall t \in \mathcal{T} \tag{7}$$

$$x_{it} \leq y_{it}, \quad \forall i \in \mathcal{N},\ \forall t \in \mathcal{T} \tag{8}$$

$$\sum_{t=1}^{T} x_{it} \leq 1, \quad \forall i \in \mathcal{N} \tag{9}$$

$$x_{it} \in [0,1],\ y_{it} \in \{0,1\},\ \forall i \in \mathcal{N},\ \forall t \in \mathcal{T}. \tag{10}$$

To achieve the starting solution (Y', X') of P_{GMKPS}, the resulting problem P'_{GMKPS} is solved; that is resolved with a Truncated-Mixed Integer Programming (noted T-MIP), where the Cplex solver is used.

Phase 2. Herein, all integral values related to Y are fixed, according to the solution reached by the first phase above. Therefore,

- Let α be the setup cost related to overall families such that $\alpha = \sum_{t=1}^{T}\sum_{i=1}^{N} f^{\star}_{it} y'_{it}$.
- Let β be the capacity consumption related to the family i such that $\beta_i = s^{\star}_i y'_{it}$, $\forall t \in \mathcal{T}$.

Thus, the reduced model P^{Red0}_{GMKPS} can be written as follows:

$$P^{Red0}_{GMKPS}: \quad \text{maximize} \quad \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{n_i} p_{ijt} x_{ijt} + \alpha \tag{11}$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\Big(\sum_{j=1}^{n_i} w_{ij} x_{ijt} + \beta_i\Big) \leq R_t, \quad \forall t \in \mathcal{T} \tag{12}$$

$$x_{ijt} \leq y'_{it} \quad , \ \forall\ j \in \{1, ..., n_i\},\ \ \forall i \in \mathcal{N},\ \forall t \in \mathcal{T} \tag{13}$$

$$\sum_{t=1}^{T} x_{ijt} \leq 1 \quad \forall i \in \mathcal{N}.\ \forall\ j \in \{1, \ldots, n_i\} \tag{14}$$

$$x_{ijt} \in \{0,1\}, \forall\ j \in \{1, ..., n_i\},\ \forall i \in \mathcal{N},\ \forall t \in \mathcal{T}. \tag{15}$$

Let $\underline{X}$ be the optimal solution of $\mathrm{P}^{Red0}_{GMKPS}$, and (X^0, Y^0) denote the (starting) solution of P_{GMKPS} such that $(X^0, Y^0) = \big(\underline{X}, Y'\big)$.

Algorithm 2 summarizes the main steps of the procedure described above, which (i) reach a starting configuration for P_{GMKPS}, and (ii) a complete feasible solution of the proposed method, as we will see in the rest of the paper.

Algorithm 2 GMKPS's starting procedure

1: **Input.** An instance of GMKPS.

2: **Output.** A starting solution for GMKPS.

3: Solve P'$_{GMKPS}$ by applying T-MIP and let (X', Y') be the achieved solution.

4: Set $\alpha = \sum_{t=1}^{T}\sum_{i=1}^{N} f^{\star}_{it} y'_{it}$, and $\beta_i = s^{\star}_i y'_{it},\ \forall t \in \mathcal{T}$.

5: Solve t$\mathrm{P}^{Red0}_{GMKPS}$ by calling T-MIP, and let $\underline{X}$ be its optimal solution.

6: **return** (X^0, Y^0), where $X^0 = \underline{X},\ Y^0 = Y'$, and Z^0 its objective value.

3.3. A basic local branching

Local Branching (LB) is a specialized optimization technique that is used as an alternative to exact methods for solving hard combinatorial optimization problems. LB has been first proposed in Fischetti & Lodi (2003), where its goal is to mimics an optimal resolution of mixed integer programming problems (MIP). Such a technic was successfully used for solving a series of combinatorial optimization (Akeb et al., 2011). The basic LB uses the principle of the branching conditions that are expressed through a series of linear inequalities. Indeed, given the following MIP:

$$\max \quad c^T x \tag{16}$$

$$Ax \leq b \tag{17}$$

$$x_j \geq 0 \qquad \forall j \in G,\ x_j \text{ integer} \tag{18}$$

$$x_k \geq 0 \qquad \forall k \in C \tag{19}$$

$$x_i \in \{0,\ 1\} \quad \forall i \in B \neq 0, \tag{20}$$

where $N = \{1, ..., n\}$ is partitioned into the following sets (B, G, C) such that B is the set of binary variables, and G and C denote the sets of integer and continuous variables, respectively. Let $\overline{x}$ be a starting solution, considered as the reference solution of MIP, and $k \in N^\star$ be the k_{Opt} neighborhood related to $\overline{x}$ corresponds to the set of feasible solutions of MIP which satisfies the following additional local (branching) constraint:

$$\Delta(x, \bar{x}) := \sum_{j \in S} (1 - x_j) + \sum_{j \in B \setminus S} x_j \leq k, \tag{21}$$

where $S = \{j \in B \mid \bar{x} = 1\}$ and the term of the left-side of inequality (21) is the number of binary variables switching their values, according to $\overline{x}$, either from 1 to 0 or from 0 to 1. Assume that the cardinality of the binary set B is fixed. Then, the following additional constraint is considered:

$$\Delta(x, \bar{x}) := \sum_{j \in S} (1 - x_j) \leq k', \tag{22}$$

where $k' = \frac{k}{2}$ (halved). Adding constraint (22) represents a branching criterion within an enumerative scheme for a MIP. In this case, according to both configurations, the space of feasible solutions related to the current branching node can be divided according to the following two complementary constraints:

$$\Delta(x, \bar{x}) \leq k \qquad \text{or} \qquad \Delta(x, \bar{x}) \geq k + 1, \tag{23}$$

where k is often provided experimentally. According to the value assigned to k in inequalities (23), the search process may be iterated by altering between normal and local branches.

A local branch is related to a complete resolution prior to branching for solving the current problem with a normal branch. Whenever a new enhancement is reached (in the local branch), local branching can iterate the resolution with the new achieved solution, where a new constraint is added to the remaining normal branch. Following the same branching principle, the last node is divided again in two branches by adding two new disjunctive constraints, i.e.,

$$\Delta(x, \bar{x}) > k, \quad \Delta(x, \bar{x}') \leq k \quad \text{(local branch)}, \tag{24}$$

and

$$\Delta(x, \bar{x}) > k, \quad \Delta(x, \bar{x}') \geq k + 1 \quad \text{(normal branch)}, \tag{25}$$

where $\bar{x}'$ denotes the new solution reached in the local branch.

The above search process is iterated until reaching a final solution which can be considered either as an optimal solution if all local branches are exactly solved or as an approximate solution if some stopping criteria are used for curtailing the search process.

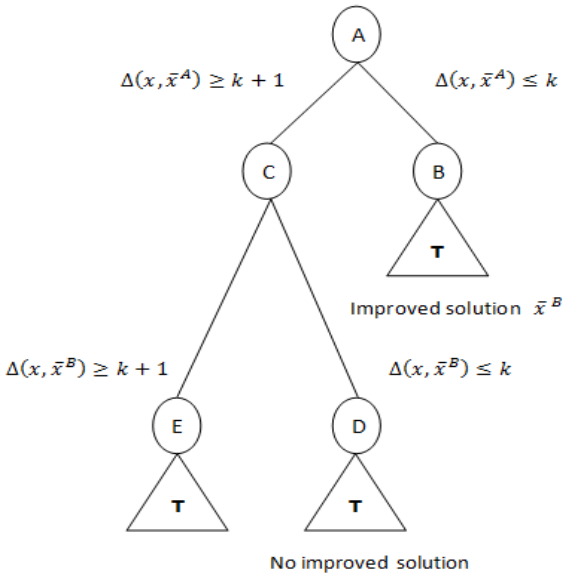

Figure 1: A local branching-based strategy

Fig 1 illustrates LB, where the triangles marked by “T" (for Tactical) corresponds to the branching subtrees to be explored through a standard “tactical" branching criterion. The starting solution $\bar{x}^A$ is an incumbent solution assigned to the root node. The right-branch

(node B) corresponds to the optimization within the k_{Opt} neighborhood $\nu(\bar{x}^A, k)$, that is performed through a tactical branching scheme covering (hopefully in short computing time) to an optimal solution in the neighborhood, say $\bar{x}^B$. Whenever the last provided solution is improved, then it becomes as a new incumbent solution. The scheme is then repeated to the left-branch (node C), where the exploration of $\nu(\bar{x}^B, k) \backslash \nu(\bar{x}^A, k)$ on the left-branch (added to node D) provides a non-improved solution. The same process is iterated till solving all subtrees hopping to reach a final solution which can be considered either as an optimal solution or as a "near-optimal" solution (as described above).

3.4. VND with a learning strategy

Most local searches use a single or two neighborhoods for improving the solution at hand. Often, both 2-opt and 3-opt operators are considered as the more simplistic operators which can be easily implemented for exploring the whole space search. Other sophisticated methods may apply a decomposition strategy on the search space for (i) delimiting certain parts and (ii) to tighten both lower and upper bounds used, hopping to accelerate the search process (for more details, the reader can refer to Brimberg et al. (2000), and Hanafi et al. (2009)).

Herein, we propose a new exploration of the search space such that the constraints used by LB are employed for exploring the whole space. Such a version can also be viewed as a special VND, where successive branches are added to successive local branches for highlighting the solutions at hand. Indeed, on the one hand, one can observe that values assigned to k (used by LB) may vary without searching for a tailored adjustment value for LB's convergence. On the other hand, such a variation on k is of great help to the VND since it uses a series of neighborhoods that coincide with the branching constraints in LB. We are going non to discuss how both LB and VND can cooperate for highlighting the solutions of GMKPS.

3.4.1. Constraints as neighborhood structures

Let (X', Y') be a feasible reference solution of P_{GMKPS} provided by the constructive method (cf. Algorithm 2). Let S_1 and S_0 be the sets related to Y' containing elements fixed to one and zero respectively, i.e.,

$$S_1 = \{(i,t), y'_{it} = 1\} \quad \text{and} \quad S_0 = \{(i,t), y'_{it} = 0\}.$$

Then, for a given nonnegative integer parameter k, we define k-opt as the neighborhood $N(Y', k)$ of the solution Y'; that is the set of the feasible solutions of P_{GMKPS} satisfying

the following additional constraint:

$$\triangle(Y, Y') = \sum_{i \in S_1} (1 - y_{it}) + \sum_{i \in S_0} y_{it} \leq k. \tag{26}$$

where the two terms on the left-hand side (of Eq. (26)) counts the number of binary variables flipping their values (according to the solution Y') either from 1 to 0 or from 0 to 1, respectively.

Because VND may also uses the parameter k for varying the neighborhood structure, from k_1 to k_{max}, we then use the same k_{max} neighborhoods. Indeed, let $N = \{N_{k_1}, ..., N_{k_{max}}\}$ be the set of structures such that each structure $N_{k_i},\ k = 1, \ldots, k_{max}$ corresponds to the current problem P to solve augmented with the local constraint related to the current LB, i.e., using Y as (a part of) the incumbent solution with its neighborhood $N(Y, k)$.

According to the above description, the following steps can be used for summarizing the principle of the approach:

1. Define N as the set of neighborhood structures, where N is reordered according to the order of exploration which will be applied by the method.
2. Apply VND, where the exploration is used according to the established order of neighborhoods on N.
3. The search process jump from the next neighborhood structure whenever LB is not able to improve the current incumbent solution.
4. The last two steps are iterated either all exploring the k_{max} neighborhood structures or whenever the runtime limit is matched.

Because LB needs a model to explore by injecting a series of valid and invalid constraints, we then describe the model used by the learning strategy. Let RP_{GMKPS} be the resulting program induced from P_{GMKPS} by relaxing overall binary variables, i.e., setting $x_{ijt} \in [0, 1]$, and $y_{it} \in [0, 1], \forall\ j = 1, \ldots, n_i,\ \ \forall\ i \in \mathcal{N},\ \forall\ t \in \mathcal{T}$ such that RP_{GMKPS} is given as follows:

$$\mathrm{RP}_{GMKPS}: \quad \max \quad \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{n_i} p_{ijt} x_{ijt} + \sum_{t=1}^{T} \sum_{i=1}^{N} f_{it} y_{it} \tag{27}$$

$$\text{s.t.} \quad \sum_{t=1}^{T} \sum_{i=1}^{N} \Big(\sum_{j=1}^{n_i} w_{ij} x_{ijt} + s_i y_{it} \Big) \leq R_t,\ \forall t \in \mathcal{T} \tag{28}$$

$$x_{ijt} \leq y_{it}, \quad \forall i \in \mathcal{N},\ \forall j \in \{1, ..., n_i\},\ \forall t \in \mathcal{T} \tag{29}$$

$$\sum_{t=1}^{T} x_{ijt} \leq 1, \quad \forall i \in \mathcal{N},\ \forall j \in \{1, ...n_i\} \tag{30}$$

$$x_{ijt} \in [0,1],\ y_{it} \in [0,1],\ \forall i \in \mathcal{N},\ \forall j \in \{1, ..., n_i\},\ \forall t \in \mathcal{T}. \tag{31}$$

To provide a first approximate solution (X, Y) for the original problem P_{GMKPS}, we first solve RP_{GMKPS}. Indeed, in this work, that problem is resolved by using the well-known Simplex method. In this case, let d_t be the number of y_{it} of the knapsack t in the component Y with non-negative values. Therefore, one can observe that the following constraint is valid for the original problem P_{GMKPS}:

$$\sum_{i=1}^{N} y_{it} \leq d_t, \quad \forall\ t \in \mathcal{T}. \tag{32}$$

Hence, by adding constraint (32) to the original problem P_{GMKPS}, and by relaxing only the variables x_{ijt}, we establish the following program:

$$\begin{aligned}
\mathrm{R}^{Y}_{GMKPS}: \qquad \max \quad & \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{n_i} p_{ijt}x_{ijt} + \sum_{t=1}^{T}\sum_{i=1}^{N} f_{it}y_{it} \\
\text{Subject to} \quad & \sum_{i=1}^{N} y_{it} \leq d_t, \quad \forall\ t \in \mathcal{T} \\
& (2),\ (3),\ \text{and}\ (4) \\
& x_{ijt} \in [0,1],\ y_{it} \in \{0,1\},\ \forall\ i \in \mathcal{N},\ \forall\ j \in \{1, ..., n_i\},\ \forall t \in \mathcal{T}.
\end{aligned}$$

- First, fix all decision variables y_{it} with integral values in the solution related to R^{Y}_{GMKPS}.
- Second, let $\mathrm{P}^{Red1}_{GMKPS}$ be the reduced (sub)problem whose decision variables are x_{ijt}, which are related to all decision variables y_{it} fixed to 1.
- Finally, the optimal solution of $\mathrm{P}^{Red1}_{GMKPS}$ induces a completed solution for the original problem P_{GMKPS}.

According to the resulting program R^{Y}_{GMKPS}, the following steps are considered:

1. The relaxation RP_{GMKPS} is optimized by using the Simplex method. Then, we collect the current primal solution $(\overline{X}, \overline{Y})$ representing the primal configuration.
2. Let D be the vector of cardinality T representing the number of decision variables associated to $\overline{y}_{it} \neq 0$ according to each period t.
3. Solve R^{Y}_{MCKS} with LB:

$$\Delta(Y, Y') \leq k \text{ (left-branch)} \quad \text{and} \quad \Delta(Y, Y') \geq k + 1 \text{ (right-branch)},$$

by using a Truncated-Mixed integer programming solver − T-MIP (herein, the Cplex solver is used by fixing a runtime limit as a stopping condition). In this case, let (X', Y') be the achieved solution.

4. Set $\alpha = \sum_{t=1}^{T} \sum_{i=1}^{N} f_{it}^{\star} y'_{it}$, and $\beta_i = s_i^{\star} y'_{it}$, $\forall t \in \mathcal{T}$.
5. Consider $\mathrm{P}_{GMKPS}^{Red1}$ as the resulting reduced program by fixing all elements y_{it}.
6. Finally, set $\overline{X}$ as the optimal solution of $\mathrm{P}_{GMKPS}^{Red1}$. Thus, (X'', Y'') is a feasible solution of P_{GMKPS} such that $(X'', Y') = \left(\overline{X}, Y'\right)$.

Algorithm 3 VND-LB Algorithm

1: **Input.** An instance of GMKPS.
2: **Output.** A (near)optimal solution $S^{\star} = (X^{\star}, Y^{\star})$ with objective value $z^{\star}$.
3: Let $S_0 = (X_0, Y_0)$ be the initial provided solution (cf. Algorithm 2).
4: Set $S^{\star} = S_0$, $S' = S_0$, and $z^{\star} = z'_{S'} = z_{S_0}$.
5: Set `Solve` = `true` and $k = k_0$.
6: **while** (the stopping condition is not performed) **do**
7: **while** (`Solve`) **do**
8: Solve R_{GMKPS}^{Y} with LB:
 $\Delta(Y, Y') \leq k$ (left-branch); $\Delta(Y, Y') \geq k + 1$ (right-branch).
9: Let $S'' = (X'', Y'')$ be the solution reached with its objective value $z'' = z_{S''}$.
10: Set $\alpha = \sum_{t=1}^{T} \sum_{i=1}^{N} f_{it}^{\star} y'_{it}$, and $\beta_i = s_i^{\star} y'_{it}$, $\forall t \in \mathcal{T}$.
11: Solve P_{GMKPS}^{Red1}, and let $\bar{X}$ be its optimal solution.
12: Set $S'' = (X'', Y'')$ the solution: $X'' = \bar{X}$ and Y'' denotes R_{GMKPS}^{Y}'s solution.
13: Set $z'' = z_{S''}$.
14: **if** $(z'' > z^{*})$ **then**
15: Set $S^{\star} = S''$, $z^{*} = z''$.
16: Set $S' = S^{\star}$ and $k = k_0$.
17: **else**
18: `Solve`=`false`, and increment k.
19: **end if**
20: **end while**
21: Set `Solve`=`true`, remove all branches and set $S' = S^{\star}$.
22: **end while**
23: **return** $(X^{\star}, Y^{\star})$ with its objective value $z^{\star}$.

3.4.2. An overview of the proposed method

Algorithm 3 describes the main steps of the proposed method that combines VND and LB, where the local branching is used as a learning strategy. It is composed of two main loops: an internal loop (from line (7) to line (20)), and a global loop (from (6) to line (22)). First, the algorithm starts by calling Algorithm 2 for computing the first incumbent

solution. Next, the internal loop (from line 7 to line 20) represents the iterative solve, where the relaxed problem R^{Y}_{GMKPS} is first solved with a T-MIP with the additional constraint $\Delta(Y, Y') \leq k$; in this case, a runtime limit is fixed to an equivalent of $3 \times T \times N$ for T-MIP (that is a value fixed before several tunings and which provides a balance between the final reached bounds and the final runtime). Herein, a feasible solution (X'', Y''), with objective value z'', is provided (line 13), where its component X'' is the result of the reduced model P^{Red1}_{GMKPS} (line 11) while the component Y'' is related to the solution of the relaxed problem P'$_{GMKPS}$ (line 8). On the one hand, the solution (X'', Y'') at hand is stored in $(X^{\star}, Y^{\star})$ and (X', Y') (lines 15 and 16) whenever its objective value z'' is better than $z^{\star}$ (related to the best solution $(X^{\star}, Y^{\star})$ find so far), and the search process continues with the restarted neighborhood (line 16). On the other hand, the provided solution is not improved and so, the internal loop is stopped (line 21), and the algorithm restarts the internal loop with the best solution $(X^{\star}, Y^{\star})$ and with a new neighborhood. Such a processes is iterated till matching the stopping condition of the global loop; that is fixed to a global number of iterations related to the size of the final neighborhood (herein, k_0 was fixed to 2 for representing a 2-opt operator while the global stopping condition k_{max} was fixed to 7; of course, we tried to find a balance between the quality of the solutions reached and the average runtimes consumed, as discussed in the experimental part in Section 4).

4. Computational results

To evaluate the effectiveness of the proposed method on benchmark instances, three sets, where each set contains 120 instances (these instances were extracted from Adouani et al. (2020)). Their optimal objective values are known([1]), and were generated according to a standard generator used in Adouani et al. (2020). The value of T varies in the discrete interval $\{5, 10, 15, 20\}$, the value of N belongs to the discrete interval $\{10, 20, 30\}$, the number of items of each period n_i varies the discrete intervals $[40, 60]$, $[60, 90]$ and $[90, 110]$ according to three sets. The used generator is summarized in what follows (according to the couple (N, T)):

- The profits and weights were generated as follows: w_{ij} belongs to the discrete interval $[10, 10000]$ while $p_{ijt} = w_{ij} + e_0$ such that e_0 is uniformly distributed in $[0, 10]$.

[1] These instances can be downloaded from https://goo.gl/zK6yZn

- $s_i = \sum_{j=1}^{n_i} a_{ij}$, $f_{it} = \sum_{j=1}^{n_i} -c_{ijt} \times e$, and $R_t = \frac{1}{2}\Big(\max_{1 \leq i \leq N} \Big\{ \sum_{j=1}^{n_i} a_{ij} + e_1 \Big\} \Big)$, where $e \in [0.15, 0.25]$ and $e_1 \in [0, \sum_{j=1}^{n_i} a_{ij}]$.

We note that the proposed method was coded in C and run, with the Cplex solver, on the Intel Pentium Core i3 with 2 GHz.

4.1. Quality of the starting reference solution

In the preliminary study, the behavior of Algorithm 2 (noted Start) used for providing the starting reference solution. This algorithm is evaluated on overall instances of the three sets described above (containing 450 instances).

On the one hand, Table 1 reports the results achieved by Start, where columns 1, 2 and 3 report the instance information, column 4 (resp. column 5) displays the average objective value (resp. average upper bound), over the ten instances of each subgroup, provided by the Cplex solver (noted z_{Cplex} and UB_{Cplex}, respectively), and column 6 reports the average objective value achieved by Start (noted z_{Start}).

On the other hand, because we try to measure the gap between the provide solution values and the upper bound, we then reported the average experimental approximation ratio of each subgroup (containing 10 instances) in Table 2; that is computed as follows, for a maximization problem.:

$$A(I) = \frac{Start(I)}{\mathrm{UB}_{Cplex}(I)},$$

where I is a given instance. Table 2 reports the average approximation ratios of Cplex (column 4 under $\mathrm{A_{Cplex}}$), and Start (column 5 under $\mathrm{A_{Start}}$).

We note that because some upper bounds published in Adouani et al. (2020) are wrong (these instances are underlined and marked with a dag symbol "†" under UB_{Cplex} in Table 1 and in Appendix (Tables from 7 to 9), and since the global average bounds for each subgroup are impacted by these errors), we then re-run for one hour the Cplex solver on overall instances and reported the values providing the maximum value for the corresponding instance.

In what follows, we comment on the results of Tables 1 and 2 :

- Over all treated instances, Algorithm 2 (Start) is able to achieve a significant experimental approximation ratio (Table 2) for the three sets; that is equal to 0.976510 (the last line of Table 2) while the Cplex solver achieves an experimental approximation ration of 0,968866; it represents a gap closest to 1.008% (it means that Start performs better than the Cplex solver).

	Gr.	#Inst.	Cplex Solver		Start
			z_{Cplex}	UB_{Cplex}	z_{Start}
$n_i \in [40, 60]$	5	10	759993.3	760015.6	759994.0
		20	790345.3	793883.0	790962.7
		30	912772.4	917130.1	913951.0
	10	10	1429753.7	1495255.2	1447150.1
		20	1588624.0	1610699.5	1591542.3
		30	1876651.0	1891855.6	1878103.8
	15	10	2008365.5	2124334.8	2038597.1
		20	2280089.3	2340844.2	2287250.7
		30	2792713.1	2835280.0†	2802630.7
	20	10	2231607.3	2337954.6	2257576.8
		20	2983649.8	3103869.9†	3006293.8
		30	3649960.9	3767380.2	3678350.3
Average			1942043.8	1998208.6†	**1954366.9**
$n_i \in [60, 90]$	5	10	1166559.3	1166569.4	1166559.3
		20	1227446.3	1234969.8	1228019.1
		30	1142283.5	1149170.6	1144936.7
	10	10	2361932.7	2417207.1	2368030.8
		20	2470204.1	2508189.1	2472903.3
		30	2273894.4	2322243.2	2286305.0
	15	10	3193044.4	3375682.7†	3234314.1
		20	3652270.4	3773497.3	3673726.8
		30	3403684.0	3521135.5	3419998.1
	20	10	3404118.2	3556659.6	3424195.3
		20	4743392.9	5027681.3†	4798403.0
		30	4373895.8	4708155.2†	4504401.4
Average			2784393.8	2896763.4†	**2810149.4**
$n_i \in [90, 110]$	5	10	1624998.0	1625413.7	1625001.9
		20	1616004.7	1616114.2	1616013.2
		30	1703710.8	1709990.5	1703833.4
	10	10	3077445.3	3187769.9	3089138.8
		20	3121688.3	3210373.0	3131089.8
		30	3325366.6	3467513.4	3393333.8
	15	10	4303341.9	4604036.1†	4366769.7
		20	4760776.8	4991858.3	4778462.7
		30	4869781.4	5245153.1	5089593.4
	20	10	4512570.3	4757056.1	4537007.5
		20	6289943.5	6672222.8†	6332641.7
		30	6472993.5	6998377.3†	6729051.5
Average			3806551.8	4007156.5†	**3865994.8**
Global Av.			2839245.07	2967376.168	**2876837.0**

Table 1: Solution quality of the starting reference solution on overall instances.

- Despite the simplicity of Start, the alternative model used by the algorithm is able to provide a good average experimental approximation ratio which varies from 0.948466 (Group $n_i \in [90, 110]$ #Inst. 15.10) and 0.999991 (Group $n_i \in [60, 90]$ #Inst. 5.10). We notice that In some cases, the average value becomes better for large instances. Hence, it can be encouraging to predict a combination of such a model with local branching-based strategy.

- Finally, Start's average global objective value, over the three sets instances (the last column and the last line of Table 1) remains interesting. Indeed, in this case, Cplex achieves an average objective value of 2839245.07 whereas the Start provides a slightly better average value of 2876837.0. .

Figure 2 illustrates the variation of the average experimental ratios provided by the

	Set	#Inst.	A_{Cplex}	A_{Start}
$n_i \in [40.60]$	5	10	0.999971	0.999972
		20	0.995544	0.996322
		30	0.995249	0.996534
	10	10	0.956194	0.967828
		20	0.986294	0.988106
		30	0.991963	0.992731
	15	10	0.945409	0.959640
		20	0.974046	0.977105
		30	0.984987	0.988485
	20	10	0.954513	0.965620
		20	0.961268	0.968563
		30	0.968833	0.976368
Average			0.971892	**0.978060**
$n_i \in [60.90]$	5	10	0.999991	0.999991
		20	0.993908	0.994372
		30	0.994007	0.996316
	10	10	0.977133	0.979656
		20	0.984856	0.985932
		30	0.979180	0.984524
	15	10	0.945896	0.958121
		20	0.967874	0.973560
		30	0.966644	0.971277
	20	10	0.957111	0.962756
		20	0.943455	0.954397
		30	0.929004	0.956723
Average			0.961209	**0.970100**
$n_i \in [90.110]$	5	10	0.999744	0.999747
		20	0.999932	0.999937
		30	0.996328	0.996399
	10	10	0.965391	0.969060
		20	0.972376	0.975304
		30	0.959006	0.978607
	15	10	0.934689	0.948466
		20	0.953708	0.957251
		30	0.928435	0.970342
	20	10	0.948606	0.953743
		20	0.942706	0.949105
		30	0.924928	0.961516
Average			0.949938	**0.964773**
Global Av.			0.968866	**0.976510**

Table 2: Variation of the average experimental approximation ratios of Start and Cplex.

reference solution on the three sets instances containing 120 instances each. The same figure shows the average experimental proximation ratios achieved by both Cplex and VND-IP. In this case, one can observe that the curve related to the results reached by Start is better than those of Cplex and VND-IP.

4.2. Performance of VND-LB

In this section, VND-LB's behavior is analyzed on the three sets containing 120 instances each (each set is divided into 12 subgroups) representing a total of 360 small, medium and large-scale benchmark instances of the literature.

First, its results are compared to those of the reference solution (provided by the first phase Start). Second and last, VND-LBs' bounds are compared to those published in Adouani et al. (2020): a Variable Neighborhood Descent with Integer Programming (noted VND-IP), and those reached by the state-of-the-art Cplex solver (noted Cplex).

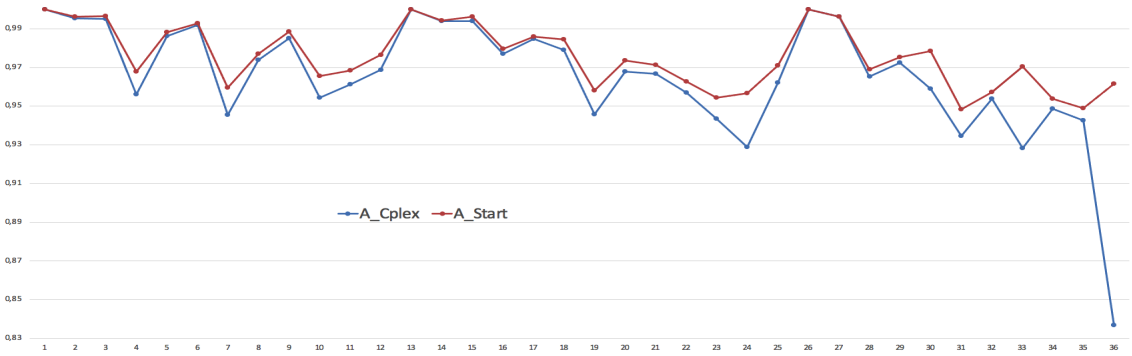

Figure 2: Variation of the average experimental approximation ratio of the starting solution compared to the Cplex's lower bound on overall subgroups.

	Gr.	#Inst.	Start without Local Branching z_{Start}	Start with Local Branching $z_{Start-LB}$
$n_i \in [40.60]$	5	10	759994	**759994.1**
		20	790962.7	790962.7
		30	913951	**913952.1**
	10	10	1447150.1	**1447155.1**
		20	1591542.3	**1591559.0**
		30	1878103.8	1878103.8
	15	10	2038597.1	**2038625.9**
		20	2287250.7	**2287263.8**
		30	2802630.7	2802630.7
	20	10	2257576.8	**2257599.1**
		20	3006293.8	**3006493.2**
		30	3678350.3	3678350.3
Average			1954366.94	**1954390.82**
$n_i \in [60.90]$	5	10	1166559.3	1166559.3
		20	1228019.1	1228019.1
		30	1144936.7	1144936.7
	10	10	2368030.8	2368030.8
		20	2472903.3	**2472903.4**
		30	2286305	**2286305.1**
	15	10	3234314.1	**3234320.8**
		20	3673726.8	**3673741.1**
		30	3419998.1	**3420001.4**
	20	10	3424195.3	3424195.3
		20	4798403	**4798403.4**
		30	4504401.4	**4504405.7**
Average			2810149.41	**2810151.84**
$n_i \in [90.110]$	5	10	1625001.9	1625001.9
		20	1616013.2	1616013.2
		30	1703833.4	1703833.4
	10	10	3089138.8	**3089142.4**
		20	3131089.8	**3131090.4**
		30	3393333.8	**3393337.9**
	15	10	4366769.7	**4366774.2**
		20	4778462.7	**4778463.6**
		30	5089593.4	**5089593.9**
	20	10	4537007.5	4537007.5
		20	6332641.7	**6332645.3**
		30	6729051.5	6729051.5
Average			3865994.78	**3865996.27**

Table 3: Effect of the local branching strategy on the iterative algorithm.

4.2.1. Effect of the local branching strategy

In this section, we study the effect of the local branching strategy when combined with the the reference solution provided by solving the mixed integer programming. We

do it by displaying, for each 12 subgroups tested (containing a total of 360 instances), by displaying the average bound achieved by both Start and VND-LB. Indeed, Table 3 reports these bounds, where column 4 displays those of Start without local branching (under z_{start}) and column 5 tallies those of VND-LB (Start with Local Branching - under $z_{start-LB}$). Now, we comment on the results of Table 3:

1. One can observe that VND-LB is able to provide a better average bounds for the three sets of instances, i.e, VND-LB achieves an average bound of 1954390.82 for the first set with $n_i \in [40.60]$ (resp. 2810151.84 for $n_i \in [60.90]$, and 3865996.27 for $n_i \in [90.110]$) while Start reaches the average value of 1954366.94 (resp 2810149.41, and 3865994.78).
2. Over all tested subgroups, VND-LB dominates Start in 8 occasions over the ten subgroups of the first set with $n_i \in [40.60]$ (resp. in 7 occasions over the ten subgroups for the two other sets).

This first study shows the positive effect of the local branching in the research process. Of course, in order to confirm this study, we will later carry out a statistical analysis for comparing the behavior of both Start and VND-LB.

4.2.2. Comparing VND-LB with other methods

Because the Cplex solver is a specialized optimal method, we then tested it using two tunings, where each version was fixed to one hour: (i) automatic search method and, (ii) dynamic search; for each of these versions, the RINS heuristic was fixed to 100); thus, the best objective value achieved by these versions are retuned as the best solution value; the RINS heuristic was setting equal to 100 for these versions respectively); in this way, these versions provides the best objective value as the best solution value of Cplex. We note that all results were extracted from Adouani et al. (2020) and we corrected their upper bounds when necessary (as underlined above: the wrong upper bounds of each instance is underlined and marked with the symbol "†"); we therefore analyze the results provided by VND-LB to those reached by the aforementioned algorithms.

Table 4 reports the average values provided by Cplex, VND-IP and VND-LB on the three sets of instances (containing 12 subgroups). Columns from 1 to 3 (on the left-side) report the instance's informations, columns 4 and 5 tally both upper and lower bounds reached by the Cplex solver for thees instances, column 6 displays the bound ($z_{\text{VND-IP}}$) extracted from Adouani et al. (2020) and column 7 displays the objective value achieved by

the proposed method VND-LB ($z_{\text{VND-LB}}$). For each set of instances, the last line displays the average values, over the 120 instances of the set, according to each method.

	Gr.	#Inst.	Cplex Solver		VND-IP	VND-LB
			z_{Cplex}	UB_{Cplex}	z_{VND-IP}	z_{VND-LB}
$n_i \in [40, 60]$	5	10	759993.3	760015.597	**759994.4**	759994.1
		20	790345.3	793882.96	790961.3	**790962.7**
		30	912772.4	917130.065	913951.0	**913952.1**
	10	10	1429753.7	1495255.17	1447095.6	**1447155.1**
		20	1588624	1610699.51	1591549.1	**1591559.0**
		30	1876651	1891855.65	**1878103.8**	**1878103.8**
	15	10	2008365.5	2124334.77	2038186.2	**2038625.9**
		20	2280089.3	2340844.23	2287007.8	**2287263.8**
		30	2792713.1	2835280.01	2801237.3	**2802630.7**
	20	10	2231607.3	2337954.61	**2257610.9**	2257599.1
		20	2983649.8	3103869.94	3003850.7	**3006493.2**
		30	3649960.9	3767380.2	3678277.5	**3678350.3**
Average			1942043.8	1998208.6	1953985.5	**1954390.8**
$n_i \in [60, 90]$	5	10	**11166559.3**	1166569.37	1166558.0	**1166559.3**
		20	1227446.3	1234969.8	**1228019.1**	**1228019.1**
		30	1142283.5	1149170.63	**1144936.7**	**1144936.7**
	10	10	2361932.7	2417207.15	2367910.5	**2368030.8**
		20	2470204.1	2508189.08	2472900.0	**2472903.4**
		30	2273894.4	2322243.19	2286305.0	**2286305.1**
	15	10	3193044.4	3375682.75	3232684.8	**3234320.8**
		20	3652270.4	3773497.32	3673529.4	**3673741.1**
		30	3403684	3521135.53	3419810.9	**3420001.4**
	20	10	3404118.2	3556659.62	3424059.9	**3424195.3**
		20	4743392.9	5027681.29	4794759.1	**4798403.4**
		30	4373895.8	4708155.22	4502885.9	**4504405.7**
Average			2784393.8	2896763.4	2809529.9	**2810151.8**
$n_i \in [90, 110]$	5	10	1624998	1625413.7	**1625001.9**	**1625001.9**
		20	1616004.7	1616114.24	**1616013.2**	**1616013.2**
		30	1703710.8	1709990.54	**1703833.4**	**1703833.4**
	10	10	3077445.3	3187769.89	3089097.3	**3089142.4**
		20	3121688.3	3210372.97	3131083.9	**3131090.4**
		30	3325366.6	3467513.42	3394182.1	**3393337.9**
	15	10	4303341.9	4604036.05	4356089.0	**4366774.2**
		20	4760776.8	4991858.27	4778300.3	**4778463.6**
		30	4869781.4	5245153.14	5089501.9	5089593.9
	20	10	4512570.3	4757056.07	**4538967.5**	4537007.5
		20	6289943.5	6672222.81	6331424.1	**6332645.3**
		30	6289943.5	6998377.31	6726455.3	**6729051.5**
Average			3791297.6	4007156.5	3864995.8	**3865996.3**

Table 4: The average objective values achieved by Cplex, VND-IP and VND-LB on the three sets.

Next, we comment on the results of Table 4, where we compare the (average) lower bounds reached (over the three sets of instances) by the proposed method VND-LB to those reached by the other methods.

1. VND-IP versus Cplex: one can observe that, in 35 occasions over the 36 groups, VND-IP outperforms the Cplex solver while its fails only in one occasion.

2. VND-LB versus Cplex: VND-LB dominates the Cplex in 35 occasions over the 36 groups, and it matches the average lower bound reached by Cplex in one occasion. In this case, VND-LB realizes a percentage of 94.44% of the better average lower bounds.

3. VND-LB versus VND-IP: on the one hand, 26 better average lower bounds are provided by VND-LB (values, in bold-space), its fails in 4 occasions, and it matches in 6 occasions the rest of values. On the other hand, the total average results achieved by VND-LB is greater than that achieved by VND-IP; indeed, it reaches an average global value of 1954390.817, 2810151.842 and 3865996.267 for set 1, 2 and 3 respectively (as observed in column 7, under $z_{\text{VND}-\text{LB}}$) whereas VND-IP provides an average value of 1953985.47, 2809529.9 and 3864995.8 for set 1, 2 and 3 respectively (column 6, under $z_{\text{VNS}-\text{IP}}$). Thus, VND-LB is able to generate 72.22% of new bounds and it matches 27.77% of the rest of the bounds.

	Gr.	#Inst.	VND-IP	VND-LB
$n_i \in [40, 60]$	5	10	58.62	55.4
		20	53.24	56.4
		30	30.29	56.30
	10	10	274.41	318.70
		20	121.08	324.10
		30	77.83	299.50
	15	10	533.15	362.00
		20	308.29	347.20
		30	262.24	315.70
	20	10	222.64	304.60
		20	559.77	383.00
		30	198.06	345.30
Average			224.97	264.01
$n_i \in [60, 90]$	5	10	68.48	63.50
		20	49.08	86.40
		30	33.88	135.50
	10	10	176.94	300.60
		20	102.21	332.50
		30	95.82	354.80
	15	10	516.53	361.90
		20	275.67	358.50
		30	320.25	355.10
	20	10	332.23	275.60
		20	544.07	442.60
		30	496.41	329.60
Average			250.96	283.04
$n_i \in [90, 110]$	5	10	40.81	92.5
		20	57.33	104.80
		30	47.73	141.60
	10	10	209.98	325.50
		20	132.15	334.00
		30	119.33	345.90
	15	10	780.88	465.50
		20	277.81	345.50
		30	216.87	366.30
	20	10	262.99	296.10
		20	515.60	408.40
		30	434.86	381.70
Average			258.03	300.65

Table 5: Variation of the average runtimes related to each subgroup for VND-IP and VND-LB.

According to the above observation (Table 4), in term of relative improvement (gap), VND-LB provides a global average gap of 1000.44 when compared to VND-IP's average value, and it becomes more significant when compared to that reached by the Cplex solver

(it becomes equal to 74698.67).

We can observe that for the average runtimes consumed by both VNS-IP and VND-LP remains difficult to do. We then tried to make an indirect comparison, and as VND-IP is a stochastic algorithm, we readjusted its average execution time for twelve trials (often, for stochastic methods, at least thirteen trials should be considered for picking the best solution value with the global average value for each instance), where the results reported in Adouani et al. (2020) represents the best solution value over all trials considered. In this case, Table 5 reports the average runtime related to each subgroup for VND-IP and VND-LB. Globally, both methods consume a closest average runtimes, even if the VND-LB consumes slightly more global average runtime.

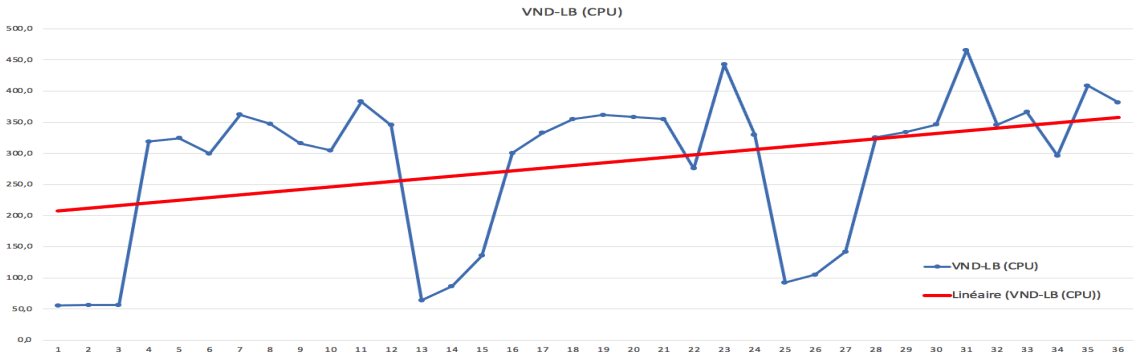

Figure 3: Variation of VND-LB's average time on overall subgroups of benchmark instances.

From the aforementioned table, Figure 3 illustrates the variation of the average runtimes consumed by VND-LB for achieving the results reported in Table 4, on the 36 subgroup of instances. In this case, one can observe that VND-LB remains competitive, overall VND-LB's runtime is not an exponential function depending on size of the instance to be solved, but it has the appearance of a linear function. We believe that VND-LB can be improved by some preprocessing procedures, such as those used for the so-called single knapsack problem, where some decision variables must be set to optimality before starting the resolution of the reduced problem.

4.2.3. A statistical analysis

Second, in order to evaluate the behavior of the reference solution Start, VND-IP, Cplex solver and the proposed VND-LB, we propose a statistic analysis using the *sign test* and the *Wilcoxon signed-rank test* statistics. The sign test is based on the number of non-negative (> 0) and negative (< 0) gaps (related to the lower bounds achieved by each algorithm) whereas the Wilcoxon signed-rank test is considered as an alternative study to

determine whether two dependent samples were selected from populations with the same distribution. Herein, we should determine if an algorithm, noted A, provides better bounds than another one, noted B (we then inverted the sign of the test). Indeed, let Sol_{A} and Sol_{B} denote their achieved values), respectively. Then, we can set (i) the hypothesis H_0: $Sol_{\mathrm{A}} - Sol_{\mathrm{B}} = D$ to express that algorithm A performs better than B (a maximization problem) and, (ii) the hypothesis H_a: algorithm B outperforms A to express the rejection of the hypothesis H_0. Thus, we can consider that the higher the average bound and the greater the number of better bounds, the better the corresponding algorithm.

Table 6: p-values for Sign test and Wilcoxon rank test on overall tested instances with the significance level $\alpha = 5\%$;

			Start vs		
	VND-IP vs Cplex	VND-LB vs Cplex	VND-IP	VND-LB	VND-IP vs VNS-LB
p-value (sign test)	< 0.0001	< 0.0001	< 0.0001	**1.000**	**1.000**
N^+	**302**	**304**	**148**	0	14
$N^=$	53	55	194	304	191
N^-	5	1	18	**56**	**155**
p-value (Wilcoxon test)	< 0.0001	< 0.0001	< 0.0001	**1.000**	**1.000**

Table 6 shows the statical analysis on overall instances by using both the *sign test* and the *sign rank test* (the detailed results containing the average bounds of VND-LB are reported in Appendix (Tables from 7 to 9). Columns from 2 to 6 display the statistical results between VND-LB, VND-IP, Cplex and Start: line 1 (resp. line 4) tallies the p-value corresponding to the sign test (resp. rank sign test) and line 2 reports the number of times that the first algorithm dominates the second one (resp. line 3 reports the number of times that the first algorithm is dominated by the second one).

From Table 6 (resp. Tables 3 and 4), one can observe what follows:

- VND-IP vs Cplex: the p-value related to the sign test (resp. Wilcoxon signed-rank test) is smallest to the significance level $\alpha = 0.05$, indicating that VND-IP performs better than Cplex (accepting the hypothesis H_0). Throughout all instances, the number of occasions that VND-IP, when compared to Cplex, matches the bounds (the values related to N^- and N^+ under VND-IP vs Cplex) is equal to 302 and 5 while in 53 occasions both methods match the same lower bound.

- VND-LB vs Cplex: VND-LB has a better behavior than that of Cplex. Indeed, the p-value of the sign test and Wilcoxon rank test are smallest than 0.0001; *i.e.* the hypothesis H_0 is approved. In this case, VND-LB outperforms the Cplex solver in 304 occasions (N^+), it matches the the same bounds in 55 occasions, and it fails in one occasion (N^-).

- Start vs VND-IP: Start remains competitive when compared to VND-IP. In this case, the p-value related to both sign test and Wilcoxon test is smallest to the significance level $\alpha = 0.05$, which means that Start dominates VND-IP (accepting the hypothesis H_0). In this case, Start provides 148 new bounds (N^+), it matches 194 bounds, and it fails in 18 occasions. In term of percentage, Start is able to realize 89.16% of new (better) bounds (out of the matched bounds) when compared to those reached by VND-IP.

- Start vs VND-LP: VND-LB has a better behavior than that of Start. Indeed, the p-value related to both sign test and Wilcoxon test is greatest than to the significance level $\alpha = 0.05$; it means that VND-LB performs better than Start (rejecting the hypothesis H_0). In this case, VND-LB provides 56 new bounds (N^+) and it matches 304 bounds. Overall instances tested, VND-LB provides a percentage of 15.56% of improved bounds.

- VND-LB vs VND-IP: VND-LB is very competitive when compared to VND-IP for overall instances. Indeed, the p-value related to both sign test and Wilcoxon test is greatest to the significance level $\alpha = 0.05$, indicating that VND-LB performs better than VND-IP (rejecting the hypothesis H_0). In this case, VND-LB is able to achieve 155 new bounds (N^-), it matches 191 bounds, and it fails in 14 occasions. We note that in term of percentage, the proposed VND-LB discovers 43.05% of new bounds.

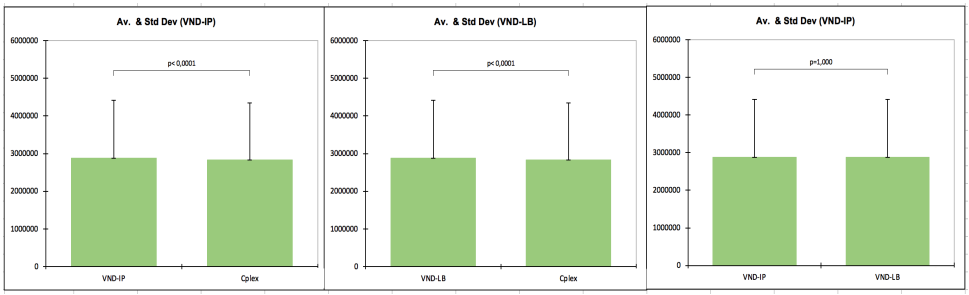

Figure 4: Illustration of the p-value related to (a) VNS-IP vs Cplex, (b) VND-LP vs Cplex, and (c) VNS-IP vs VND-LP.

Figure 4 shows the variation of the p-values related to both Sign test and Wilcoxon signed-rank test when comparing each pair of algorithms, i.e., (VND-IP, Cplex), (VND-LB, Cplex) and (VND-IP, VND-LB). First, Figure 4.(a) illustrates the superiority of VND-IP when compared to Cplex, while the overall average values achieved for VND-IP are better (see Table 4, line Average, column under $z_{\text{VND}-\text{IP}}$). Second, Figures 4.(b) shows the

superiority of VND-LB when compared to Cplex (see Table 4, line Average, column under $z_{\text{VND-LP}}$). Third and last, Figure 4.(c) illustrates the p-value related to the hypothesis $H_0 : Sol_{\text{VND-IP}} - Sol_{\text{VND-LB}} > 0$; in this case, one can observe that the hypothesis H_0 is rejected, which means that VND-LB dominates VND-IP.

Finally, Figure 5.(a) illustrates the variation of the average gap achieved by VND-LB which is related to $\text{A}_{\text{VND-LB}} - \text{A}_{VND-IP}$ (normalized). Globally, one can observe that the average gaps are much better for VND-LB than those achieved by VND-IP, which means that VND-LB is a competitive method for the problem studied in this paper. In this case, VND-LB achieves 23 non-negative values over the 36 subgroups; in term of percentage, it represents a percentage of 63.89% of the best average gaps.

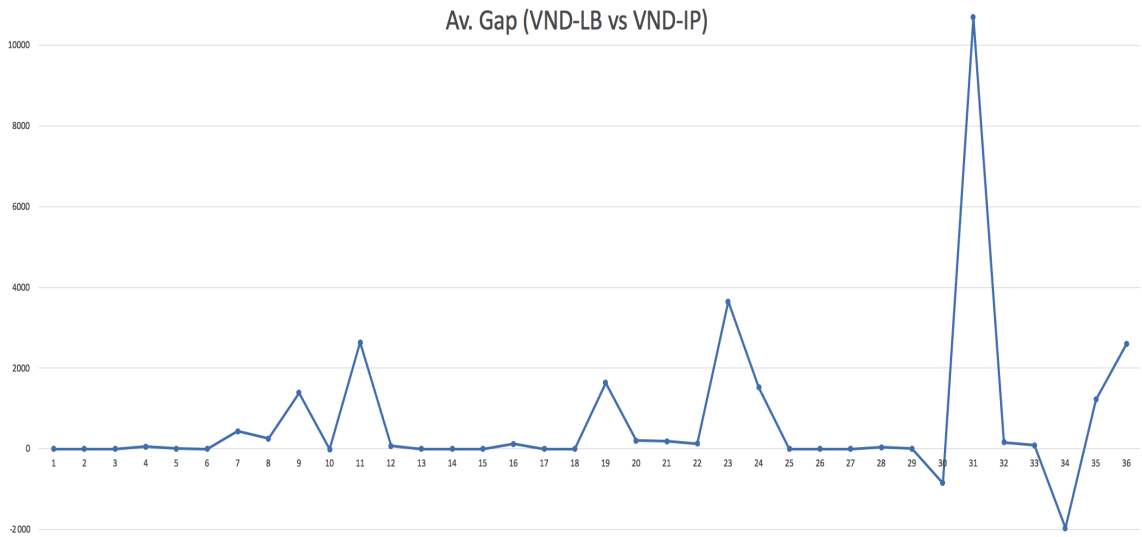

Figure 5: Variation of the average gap related to both VND-LB and VNS-IP on (all) subgroups.

5. Conclusion

In this paper the generalized multiple-choice knapsack problem with setup is solved with an iterative descent method. The main idea used is based upon solving a series of reduced subproblems built by adding a series of cardinality constraints. First, a starting reference solution was built by solving a special mixed integer programming. Second, a learning strategy using branching constraints was applied for iteratively highlighting the quality of the solutions ate hand. The method mimics a variable descent method combined with local branching. Finally, the designed method was experimentally evaluated on benchmark instances of the literature, and its obtained results were compared to those reached by a

recent method of the literature and the state-of-the-art Cplex solver. New bounds have been discovered.

As the studied problem belongs to the knapsack problem family, there are plenty possibilities for further investigation involving efficient methods able to enhance the proposed method. First, we believe that some preprocessing procedures can be tailored for the studied problem, where tights bounds and valid constraints can be added for reducing the size of initial instance. Such a method may be viewed as hybrid method, where a neighborhood decomposition procedure can be combined with a fix and solve procedure. Second, Benders decomposition may be applied for the studied problem, where both types of variables may cooperate for providing an iterative method based upon injecting successive Benders constraints. Finally, we believe that a parallel approach may be designed as another direction of research. In this case, because several neighborhoods are used to intensify the search process, we believe that a cooperative parallel method can improve the quality of the solutions while reducing the runtime of the method.

References

Adouani, Y., Jarboui, B., & Masmoudi, M. (2019). A variable neighborhood search with integer programming for the zero-one multiple-choice knapsack problem with setup. In A. Sifaleras, S. Salhi, & J. Brimberg (Eds.), *Variable Neighborhood Search* (pp. 52–166). Cham": Springer.

Adouani, Y., Jarboui, B., & Masmoudi, M. (2020). Efficient matheuristic for the generalized multiple knapsack problem with setup. *European Journal of Industrial Engineering*, *14*, 715–741. doi:10.1504/EJIE.2020.109906.

Akeb, A., Hifi, M., & Ould Ahmed Mounir, M. (2011). Local branching-based algorithms for the disjunctively constrained knapsack problem. *Computers & Industrial Engineering*, *60*, 811–820. doi:https://doi.org/10.1016/j.cie.2011.01.019.

Amiri, A. (2020). A lagrangean based solution algorithm for the knapsack problem with setups. *Expert Systems with Applications*, *143*, 113077. doi:https://doi.org/10.1016/j.eswa.2019.113077.

Boukhar, S., Dahmani, I., & Hifi, M. (2020). Effect of valid cardinality constraints in local branching: the case of the knapsack problem with setup. *Information Technology in Industry*, *8*, 8–20.

Boukhari, S., Dahmani, I., & Hifi, M. (2020). Local branching strategy-based method for the knapsack problem with setup. In *In Proceedings of the 4th International Conference on Computer Science and Information Technology* (pp. 65–75). David C. Wyld et al. (Eds) volume 10(16). doi:10.5121/csit.2020.101606.

Boukhari, S., Dahmani, I., & Hifi, M. (2022). Computational power of a hybrid algorithm for solving the multiple knapsack problem with setup. In K. Arai (Ed.), *Intelligent Computing* (pp. 154–168). Cham: Springer.

Brimberg, J., Hansen, P., Mladenovic, N., & Taillard, E. (2000). Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem. *Operations Research*, *48*, 444–460. doi:10.1287/opre.48.3.444.12431.

Chebil, K., & Khemakhem, M. (2015). A dynamic programming algorithm for the knapsack problem with setup. *Computers & Operations Research*, *64*, 40–50. doi:https://doi.org/10.1016/j.cor.2015.05.005.

Chen, M., Wu, C., Tang, X., Peng, X., Zeng, Z., & Liu, S. (2019). An efficient deterministic heuristic algorithm for the rectangular packing problem. *Computers & Industrial Engineering*, *137*, 106097. doi:https://doi.org/10.1016/j.cie.2019.106097.

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. doi:https://doi.org/10.1016/j.cor.2016.11.015.

Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming*, *98*, 23–47. doi:10.1007/s10107-003-0395-5.

Furini, F., Monaci, M., & Traversi, E. (2018). Exact approaches for the knapsack problem with setups. *Computers & Operations Research*, *90*, 208–220. doi:https://doi.org/10.1016/j.cor.2017.09.019.

Hanafi, S., Lazić, J., Mladenović, N., & Wilbaut, C. (2009). Variable neighbourhood decomposition search with bounding for multidimensional knapsack problem. (pp. 2018–2022). volume 42. doi:10.3182/20090603-3-RU-2001.0502.

Hansen, P., & Mladenović, N. (2003). Variable neighborhood search. In F. Glover, & G. Kochenberger (Eds.), *Handbook of Metaheuristics* (pp. 145–184). Boston, MA: Springer. doi:10.1007/0-306-48056-5_6.

Khemakhem, M., & Chebil, K. (2016). A tree search based combination heuristic for the knapsack with setup. *Computers & Industrial Engineering*, *99*, 280–286. doi:https://doi.org/10.1016/j.cie.2016.07.021.

Lahyan, R., Chebil, K., Khemakhem, M., & Coelho, L. (2019). Mathheristics for solving the multiple knapsack problem with setup. *Computers & Industrial Engineering*, *129*, 76–89. doi:https://doi.org/10.1016/j.cie.2019.01.010.

Liu, Q., Cheng, H., Tian, T., Wang, Y., Leng, J., Zhao, R., Zhang, H., & Wei, L. (2021). Algorithms for the variable-sized bin packing problem with time windows. *Computers & Industrial Engineering*, *155*, 107175. doi:https://doi.org/10.1016/j.cie.2021.107175.

Merkle, R., & Hellman, M. (1978). Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, *24*, 525–530. doi:10.1109/TIT.1978.1055927.

Perboli, G., Gobbato, L., & Perfetti, F. (2014). Packing problems in transportation and supply chain: new problems and trends. *Procedia - Social and Behavioral Sciences*, *111*, 672–681. doi:https://doi.org/10.1016/j.sbspro.2014.01.101.

Plata-González, L., Amaya, I., Ortiz-Bayliss, J., Conant-Pablos, S., Terashima-Marín, H., & Coello Coello, C.-A. (2019). Evolutionary-based tailoring of synthetic instances for the knapsack problem. *Soft Computing*, *23*, 12711–12728. doi:10.1007/s00500-019-03822-w.

Yang, Y. (2006). *Knapsack problems with setup*. Ph.D. thesis Industrial and Systems Engineering, Auburn University.

Appendix

In this section, we report all values related to both upper and lower bounds provided by the three compared methods. Table 7 shows all results related to the instances of the first group ($n_i \in [40, 60]$), Table 8 displays those related to the second set ($n_i \in [60, 90]$), and Table 9 tallies those related to the third set ($n_i \in [90, 110]$).

	Cplex		VND-IP	VND-LB		Cplex		VND-IP	VND-LB
Inst.	UB_{Cplex}	z_{Cplex}	z_{VND-IP}	z_{VND-LB}	Inst.	UB_{Cplex}	z_{Cplex}	z_{VND-IP}	z_{VND-LB}
I-5-10-1	812939,4	812913	812913	812913	I-15-10-1	2399935,2	2279343	2319142	2319142
I-5-10-2	958561,6	958547	958547	958547	I-15-10-2	2390757,4	2251230	2290209	2290209
I-5-10-3	960159,2	960121	960121	960121	I-15-10-3	2403221,7	2228216	2280258	2280557
I-5-10-4	676489,7	676466	676466	676466	I-15-10-4	2177210,3	2009983	2049983	2050062
I-5-10-5	955249,7	955234	955234	955234	I-15-10-5	2354597,3	2307535	2312884	2312884
I-5-10-6	669293,9	669282	669285	669285	I-15-10-6	2165591,5 †	1999374	2046672	2048590
I-5-10-7	668575,5	668540	668538	668535	I-15-10-7	1862773,3 †	1790297	1807162	1807754
I-5-10-8	612250,1	612232	612235	612235	I-15-10-8	1881900,0	1809485	1817259	1817332
I-5-10-9	685778,4	685764	685764	685764	I-15-10-9	1873553,4	1736682	1774638	1775470
I-5-10-10	600858,5	600834	600841	600841	I-15-10-10	1733807,5 †	1671510	1683655	1684259
I-5-20-1	1002339,5	1001455	1001475	1001475	I-15-20-1	2755181,0	2697518	2706826	2706826
I-5-20-2	628881,2	628864	628864	628864	I-15-20-2	1851136,5 †	1799174	1798826	1799363
I-5-20-3	674386,0	674246	674343	674357	I-15-20-3	2235899,7	2184664	2195401	2195877
I-5-20-4	928901,0	917839	918840	918840	I-15-20-4	2536908,4	2467125	2476018	2476028
I-5-20-5	936002,0	936002	936002	936002	I-15-20-5	2936926,4	2903389	2910999	2910999
I-5-20-6	798017,6	774877	779867	779867	I-15-20-6	2456596,0	2370180	2380982	2380982
I-5-20-7	920253,0	920233	920233	920233	I-15-20-7	2757730,9	2693864	2701151	2701151
I-5-20-8	682780,0	682757	682760	682760	I-15-20-8	1968416,0	1911000	1916093	1916632
I-5-20-9	591197,7	591137	591179	591179	I-15-20-9	1694740,4	1648329	1653975	1654049
I-5-20-10	776071,7	776043	776050	776050	I-15-20-10	2214906,7 †	2125650	2129807	2130731
I-5-30-1	816930,9	816917	816917	816917	I-15-30-1	2722319,7	2687640	2689634	2689997
I-5-30-2	1018124,6	1007880	1009108	1009108	I-15-30-2	2894647,8 †	2855668	2845417	2858963
I-5-30-3	1028641,3	1028614	1028614	1028614	I-15-30-3	2746848,8	2724765	2725778	2725778
I-5-30-4	1106203,3	1094667	1098280	1098280	I-15-30-4	3026950,8	3001183	3006413	3006413
I-5-30-5	832732,6	829696	829685	829696	I-15-30-5	2499499,5	2467329	2475063	2475088
I-5-30-6	1013280,1	1013233	1013233	1013233	I-15-30-6	3183465,4	3147406	3149552	3149552
I-5-30-7	765399,6	753403	760349	760349	I-15-30-7	2672791,2	2634893	2637392	2637392
I-5-30-8	905119,2	905104	905104	905104	I-15-30-8	3065731,9	2987255	3035613	3035613
I-5-30-9	753452,1	753433	753433	753433	I-15-30-9	2778551,5	2694194	2719783	2719783
I-5-30-10	931416,9	924777	924787	924787	I-15-30-10	2761993,5	2726798	2727728	2727728
I-10-10-1	1629716,7	1596969	1603954	1603964	I-20-10-1	2425354,1	2328269	2337831	2337831
I-10-10-2	1782820,3	1749331	1755536	1755536	I-20-10-2	2394577,2	2306563	2331711	2321137
I-10-10-3	1688552,0	1630853	1636663	1636678	I-20-10-3	2454323,0	2363875	2381083	2381083
I-10-10-4	1478382,8	1381215	1401813	1401915	I-20-10-4	2395021,5	2258716	2297629	2297629
I-10-10-5	1914421,3	1889523	1892365	1892365	I-20-10-5	2355104,7	2289114	2297309	2297309
I-10-10-6	1449577,6	1348515	1373297	1373303	I-20-10-6	2358599,2	2259105	2267316	2267316
I-10-10-7	1234418,8	1200768	1211432	1211591	I-20-10-7	2277787,8	2161100	2184523	2184523
I-10-10-8	1264532,2	1172875	1216286	1216369	I-20-10-8	2137761,1 †	2029634	2062592	2066994
I-10-10-9	1289770,2	1235097	1247261	1247416	I-20-10-9	2364510,6 †	2236693	2296965	2300815
I-10-10-10	1220359,8	1092391	1132349	1132414	I-20-10-10	2216506,9 †	2083004	2119150	2121354
I-10-20-1	1896294,3	1864775	1871292	1871292	I-20-20-1	3892568,3 †	3772724	3795371	3811615
I-10-20-2	1351291,4	1287727	1293785	1293791	I-20-20-2	2419631,5 †	2295290	2314756	2317185
I-10-20-3	1496848,1	1477351	1479116	1479116	I-20-20-3	2920812,9 †	2828263	2850146	2852165
I-10-20-4	1690777,4	1682637	1682672	1682672	I-20-20-4	3445949,1 †	3274439	3302637	3302838
I-10-20-5	1974743,0	1974515	1974552	1974552	I-20-20-5	3833888,1	3714273	3730447	3730445
I-10-20-6	1704503,2	1693986	1694830	1694830	I-20-20-6	3339450,7 †	3242898	3264119	3265111
I-10-20-7	1896750,2	1891434	1891514	1891514	I-20-20-7	3657873,6 †	3520490	3525277	3525731
I-10-20-8	1447937,6	1394663	1402275	1402282	I-20-20-8	2496710,4	2380725	2405080	2405269
I-10-20-9	1188892,7	1172195	1175840	1175900	I-20-20-9	2245435,6 †	2125538	2151009	2153260
I-10-20-10	1458957,2	1446957	1449615	1449641	I-20-20-10	2786378,8 †	2681858	2699665	2701313
I-10-30-1	1631703,8	1613457	1613746	1613746	I-20-30-1	3629892,3	3523333	3562054	3562337
I-10-30-2	2048699,2	2031220	2031831	2031831	I-20-30-2	3743516,9	3634824	3659987	3659987
I-10-30-3	1959136,1	1958082	1958088	1958088	I-20-30-3	3728201,3	3607676	3639456	3639457
I-10-30-4	2177876,1	2164724	2165333	2165333	I-20-30-4	3981454,9	3861462	3886415	3886415
I-10-30-5	1695739,4	1663956	1669857	1669857	I-20-30-5	3620672,3	3483875	3519183	3519196
I-10-30-6	2131073,6	2124359	2125488	2125488	I-20-30-6	4127334,0	4032184	4065645	4065645
I-10-30-7	1691681,1	1680246	1681310	1681310	I-20-30-7	3474772,1	3336973	3347683	3347703
I-10-30-8	1985227,9	1960655	1962504	1962504	I-20-30-8	3943031,5	3822736	3861978	3862174
I-10-30-9	1769772,0	1750671	1750834	1750834	I-20-30-9	3809965,8	3696808	3728062	3728089
I-10-30-10	1827647,3	1819140	1822047	1822047	I-20-30-10	3614960,9	3499738	3512312	3512500

Table 7: VND-LB versus VND-IP and Cplex solver on the first set of instances ($n_i \in [40, 60]$).

Inst.	Cplex		VND-IP	VND-LB	Inst.	Cplex		VND-IP	VND-LB
	UB_{Cplex}	z_{Cplex}	z_{VND-IP}	z_{VND-LB}		UB_{Cplex}	z_{Cplex}	z_{VND-IP}	z_{VND-LB}
I-5-10-1	1135995,1	1135990	1135990	1135990	I-15-10-1	3486717,6 †	3212770	3304096	3318511
I-5-10-2	1309362,2	1309351	1309351	1309351	I-15-10-2	3665736,6	3470872	3518744	3518744
I-5-10-3	838496,0	838484	838484	838484	I-15-10-3	2838040,9	2658053	2691244	2691129
I-5-10-4	1338935,1	1338922	1338922	1338922	I-15-10-4	3614004,3	3488463	3502706	3502706
I-5-10-5	1047794,4	1047782	1047782	1047782	I-15-10-5	3593446,7 †	3227304	3292369	3294198
I-5-10-6	1089307,5	1089295	1089295	1089295	I-15-10-6	3223946,8	3047489	3108121	3108018
I-5-10-7	1353689,3	1353672	1353659	1353672	I-15-10-7	3508713,2	3371434	3387869	3387869
I-5-10-8	1153293,9	1153287	1153287	1153287	I-15-10-8	3262349,5	3126731	3159139	3159139
I-5-10-9	869138,2	869128	869128	869128	I-15-10-9	2891922,8	2754402	2784983	2785294
I-5-10-10	1529682,0	1529682	1529682	1529682	I-15-10-10	3671949,1	3572926	3577577	3577600
I-5-20-1	1099777,0	1099777	1099777	1099777	I-15-20-1	3161322,0	2996410	3014696	3015241
I-5-20-2	1323150,2	1323122	1323130	1323130	I-15-20-2	3958086,8 †	3848426	3865508	3865787
I-5-20-3	1329784,0	1329746	1329746	1329746	I-15-20-3	4207958,5	4101826	4137427	4137427
I-5-20-4	1022396,7	996801	997810	997810	I-15-20-4	3008171,1 †	2867360	2903412	2904244
I-5-20-5	1320245,2	1293484	1293496	1293496	I-15-20-5	3861452,7	3759258	3768181	3768188
I-5-20-6	1430399,8	1417408	1422107	1422107	I-15-20-6	3846736,4	3733570	3740469	3740466
I-5-20-7	866475,3	866460	866460	866460	I-15-20-7	3207190,4	3133826	3141879	3142163
I-5-20-8	1585294,4	1575520	1575520	1575520	I-15-20-8	4795781,9	4683390	4721239	4721239
I-5-20-9	1252205,9	1252189	1252189	1252189	I-15-20-9	4034438,5	3943646	3948143	3948296
I-5-20-10	1119969,6	1119956	1119956	1119956	I-15-20-10	3653834,8	3454992	3494340	3494360
I-5-30-1	993038,3	968604	973907	973907	I-15-30-1	3405075,5 †	3272114	3282773	3282893
I-5-30-2	1154947,4	1154930	1154933	1154933	I-15-30-2	3614515,1	3565078	3569378	3569394
I-5-30-3	1179313,5	1179300	1179300	1179300	I-15-30-3	3537341,7	3434248	3448317	3448458
I-5-30-4	1045545,1	1042526	1044345	1044345	I-15-30-4	3578858,8	3461172	3477683	3477753
I-5-30-5	1178737,6	1178724	1178724	1178724	I-15-30-5	3522495,8	3370783	3393784	3393937
I-5-30-6	1284703,2	1284699	1284699	1284699	I-15-30-6	3882023,9	3756185	3763712	3763905
I-5-30-7	1260990,0	1260925	1260925	1260925	I-15-30-7	3481273,3	3319363	3341309	3341309
I-5-30-8	1136260,6	1136247	1136247	1136247	I-15-30-8	3505704,3	3423595	3435875	3435963
I-5-30-9	937821,7	896531	915938	915938	I-15-30-9	2824651,4 †	2711464	2744656	2745631
I-5-30-10	1320349,0	1320349	1320349	1320349	I-15-30-10	3859415,4	3722838	3740622	3740771
I-10-10-1	2243243,4	2204823	2204837	2204837	I-20-10-1	3524142,9	3374186	3380053	3380053
I-10-10-2	2699667,3	2641011	2644369	2644369	I-20-10-2	3669298,3	3521260	3531702	3531702
I-10-10-3	1866691,55 †	1758084	1783114	1783727	I-20-10-3	3618168,8	3372284	3429730	3429796
I-10-10-4	2845871,6	2778427	2778490	2778490	I-20-10-4	3648804,5	3497229	3518703	3518703
I-10-10-5	2342385,2	2291052	2295319	2295319	I-20-10-5	3681807,2	3492697	3532934	3532934
I-10-10-6	2002491,6	1968602	1969663	1969786	I-20-10-6	3550640,3	3372427	3372946	3372946
I-10-10-7	2788961,6	2733547	2740557	2740557	I-20-10-7	3511067,9	3418222	3418334	3418334
I-10-10-8	2465311,7	2434666	2441336	2441336	I-20-10-8	3270509,3	3200694	3202779	3202779
I-10-10-9	1953948,96 †	1896830	1897986	1898453	I-20-10-9	3406038,1 †	3202778	3244499	3245787
I-10-10-10	2963498,7	2912285	2923434	2923434	I-20-10-10	3686118,8	3589405	3608919	3608919
I-10-20-1	2099980,5	2091878	2091981	2091983	I-20-20-1	4203252,1 †	3828935	3887847	3893431
I-10-20-2	2572867,1	2528659	2532270	2532270	I-20-20-2	5369232,4 †	5125476	5128914	5151611
I-10-20-3	2869639,2	2830883	2836991	2836991	I-20-20-3	5273194,8 †	4941808	5044642	5045902
I-10-20-4	1993776,2	1954836	1960748	1960757	I-20-20-4	4256935,3 †	3914397	3949932	3951660
I-10-20-5	2620943,2	2599022	2600713	2600713	I-20-20-5	5306521,4	4968551	5021380	5021505
I-10-20-6	2598780,7	2549692	2551291	2551291	I-20-20-6	5172818,4 †	4985182	5068629	5069767
I-10-20-7	2032942,8	1997195	2000177	2000200	I-20-20-7	4416511,3 †	4114277	4150699	4151816
I-10-20-8	3262145,1	3223238	3228096	3228096	I-20-20-8	6201906,2	6038440	6114417	6114417
I-10-20-9	2647242,5	2615464	2615544	2615544	I-20-20-9	5261506,5	5019634	5053860	5054153
I-10-20-10	2383573,3	2311174	2311189	2311189	I-20-20-10	4814934,4 †	4497229	4527271	4529772
I-10-30-1	2396805,9	2349287	2352883	2352883	I-20-30-1	4643751,3	3718902	4444768	4444996
I-10-30-2	2372374,5	2314128	2327204	2327204	I-20-30-2	4978668,3	4738899	4756582	4756910
I-10-30-3	2297102,9	2257830	2262071	2262071	I-20-30-3	4580819,1 †	4331037	4374171	4376779
I-10-30-4	2222663,2	2172125	2172867	2172867	I-20-30-4	4761669,1	4523887	4602355	4602725
I-10-30-5	2333476,0	2194934	2275439	2275439	I-20-30-5	4776002,5	4499680	4535788	4535917
I-10-30-6	2551577,7	2505779	2513215	2513215	I-20-30-6	5183962,2	5015162	5046797	5047024
I-10-30-7	2318451,4	2272760	2280865	2280866	I-20-30-7	4543007,9 †	4332863	4355036	4357065
I-10-30-8	2298487,4	2278835	2278915	2278915	I-20-30-8	4638483,6 †	4382343	4408044	4408890
I-10-30-9	1908223,4	1906973	1907004	1907004	I-20-30-9	4000858,4 †	3477005	3753223	3755059
I-10-30-10	2523269,4	2486293	2492587	2492587	I-20-30-10	4974329,6 †	4719180	4752095	4758692

Table 8: VND-LB versus VND-IP and Cplex solver on the second set of instances ($n_i \in [60, 90]$).

	Cplex		VND-IP	VND-LB		Cplex		VND-IP	VND-LB
Inst.	UB_{Cplex}	z_{Cplex}	z_{VND-IP}	z_{VND-LB}	Inst.	UB_{Cplex}	z_{Cplex}	z_{VND-IP}	z_{VND-LB}
I-5-10-1	1867460,7	1867448	1867448	1867448	I-15-10-1	4663462,6	4364645	4435579	4435579
I-5-10-2	1674734,6	1674727	1674727	1674727	I-15-10-2	4658581,3	4328719	4422996	4422996
I-5-10-3	1759796,2	1759784	1759784	1759784	I-15-10-3	4826291,4	4614863	4628569	4628583
I-5-10-4	1515981,2	1515973	1515973	1515973	I-15-10-4	4703016,1 †	4299610	4411006	4418089
I-5-10-5	1731002,4	1730990	1730990	1730990	I-15-10-5	4807290,3 †	4531473	4562612	4581083
I-5-10-6	1412958,7	1408890	1408929	1408929	I-15-10-6	4612355,4 †	4248478	4321129	4345145
I-5-10-7	1692394,5	1692386	1692386	1692386	I-15-10-7	4427701,9 †	4109056	4125871	4167176
I-5-10-8	1528505,7	1528500	1528500	1528500	I-15-10-8	4556780,9 †	4325640	4369817	4372684
I-5-10-9	1530623,6	1530612	1530612	1530612	I-15-10-9	4460509,9 †	4155855	4197352	4211577
I-5-10-10	1540679,5	1540670	1540670	1540670	I-15-10-10	4324370,4	4055080	4085959	4084830
I-5-20-1	1604652,1	1604648	1604648	1604648	I-15-20-1	5383483,3	5192174	5221403	5221526
I-5-20-2	1737952,8	1737941	1737941	1737941	I-15-20-2	5234171,4	5086268	5114107	5114149
I-5-20-3	1657767,0	1657099	1657101	1657101	I-15-20-3	4733824,0	4463771	4477901	4477896
I-5-20-4	1551060,4 †	1550979	1551052	1551052	I-15-20-4	4618848,3	4394499	4411756	4411860
I-5-20-5	1378609,2	1378600	1378600	1378600	I-15-20-5	4757735,2	4484010	4490047	4490056
I-5-20-6	1657277,8	1657270	1657270	1657270	I-15-20-6	5374620,4 †	5120790	5140943	5141160
I-5-20-7	2142847,0	2142847	2142847	2142847	I-15-20-7	5528202,0	5340987	5352137	5352154
I-5-20-8	1389068,6	1388771	1388781	1388781	I-15-20-8	4602491,5	4350319	4358015	4358015
I-5-20-9	1385268,7	1385258	1385258	1385258	I-15-20-9	4502564,7 †	4227636	4244785	4245631
I-5-20-10	1656638,7	1656634	1656634	1656634	I-15-20-10	5182641,8	4947314	4971909	4972189
I-5-30-1	1537911,4	1530380	1531024	1531024	I-15-30-1	4772407,5	4602055	4612962	4612962
I-5-30-2	1516879,4	1499879	1499985	1499985	I-15-30-2	4473156,9	2779338	4231845	4231939
I-5-30-3	1543845,9	1527414	1527417	1527417	I-15-30-3	4958481,7	4178076	4811305	4811308
I-5-30-4	1721643,8	1711958	1712120	1712120	I-15-30-4	5509799,1	5335096	5350587	5350778
I-5-30-5	1703913,5	1703226	1703232	1703232	I-15-30-5	5125347,8	4973120	4985615	4985641
I-5-30-6	1622188,5	1621296	1621348	1621348	I-15-30-6	5704290,5	5537299	5551034	5551212
I-5-30-7	1960411,7	1949880	1950132	1950132	I-15-30-7	5239904,0	5037565	5050284	5050439
I-5-30-8	1956219,8	1956204	1956204	1956204	I-15-30-8	5201317,7	5017786	5026460	5026456
I-5-30-9	1846949,6	1846939	1846940	1846940	I-15-30-9	6245031,9	6153111	6165733	6165873
I-5-30-10	1689941,6	1689932	1689932	1689932	I-15-30-10	5221794,4	5084368	5109194	5109331
I-10-10-1	3298297,2	3230070	3230187	3230185	I-20-10-1	4671113,6	4432546	4456197	4452034
I-10-10-2	3183666,6	3089879	3106717	3106746	I-20-10-2	4847811,0	4619609	4628011	4628011
I-10-10-3	3645863,7	3544868	3545610	3545610	I-20-10-3	4837582,5	4592648	4628083	4628083
I-10-10-4	3046928,0	2991357	2994730	2994867	I-20-10-4	4741308,6	4530169	4571407	4555970
I-10-10-5	3464918,4	3336948	3347290	3347290	I-20-10-5	4813860,0	4579438	4598196	4598196
I-10-10-6	3024398,3	2923356	2928397	2928397	I-20-10-6	4806111,7	4563157	4572502	4572502
I-10-10-7	2987072,8	2847341	2850447	2850559	I-20-10-7	4655998,7	4389542	4421131	4421131
I-10-10-8	3276957,3	3068803	3138789	3138855	I-20-10-8	4612266,3	4379102	4421462	4421462
I-10-10-9	3023569,3	2905866	2912622	2912659	I-20-10-9	4797332,4	4532740	4554606	4554606
I-10-10-10	2926027,2	2835965	2836184	2836256	I-20-10-10	4787175,8	4506752	4538080	4538080
I-10-20-1	3422171,5	3316159	3325963	3325985	I-20-20-1	6882124,0	6524573	6607329	6607411
I-10-20-2	3221796,6	3206335	3206388	3206388	I-20-20-2	6851117,2	6515193	6540068	6540264
I-10-20-3	3099616,4	3045058	3047775	3047778	I-20-20-3	6364792,9	5952616	5990005	5990292
I-10-20-4	3052492,9	3012783	3013678	3013678	I-20-20-4	6425687,1	6026242	6060176	6060526
I-10-20-5	2885152,8	2730742	2738251	2738270	I-20-20-5	6411272,6 †	5998329	6012405	6015920
I-10-20-6	3390359,6	3168667	3225259	3225259	I-20-20-6	7429146,2	7092992	7127351	7127701
I-10-20-7	3689449,0	3617475	3618654	3618654	I-20-20-7	7251142,2	6899617	6967739	6968086
I-10-20-8	3061293,5	2980099	2980334	2980356	I-20-20-8	6199404,5 †	5790296	5816364	5820618
I-10-20-9	2875364,0	2806390	2818858	2818857	I-20-20-9	5892109,9 †	5438442	5500260	5502121
I-10-20-10	3406033,2	3333175	3335679	3335679	I-20-20-10	7015431,3 †	6661135	6692544	6693514
I-10-30-1	3197373,9	3104169	3117735	3109134	I-20-30-1	6513834,6	6164119	6198551	6198763
I-10-30-2	3072936,3	2975502	2981670	2981680	I-20-30-2	6336829,2 †	3717381	5968860	5971906
I-10-30-3	3319986,9	3211083	3214072	3214072	I-20-30-3	6726885,1 †	6401531	6425674	6447190
I-10-30-4	3449863,4	3399368	3399720	3399720	I-20-30-4	7506803,1	7233264	7282004	7282190
I-10-30-5	3304489,7	2797469	3147562	3147571	I-20-30-5	6765025,2	6476298	6499874	6500088
I-10-30-6	3679052,2	3655217	3659671	3659671	I-20-30-6	7514031,2	7270337	7283296	7283550
I-10-30-7	3442890,2	3388084	3395345	3395384	I-20-30-7	6748013,6	6425692	6466879	6467157
I-10-30-8	3579817,7	3509392	3512831	3512831	I-20-30-8	6948207,9	6644286	6687620	6687875
I-10-30-9	4049564,3	3703620	4001165	4001166	I-20-30-9	8026206,7	7820264	7850004	7850004
I-10-30-10	3579159,6	3509762	3512050	3512150	I-20-30-10	6897936,5	6576763	6601791	6601792

Table 9: VND-LB versus VND-IP and Cplex solver on the third set of instances ($n_i \in [90, 110]$).