Optimization Letters manuscript No.
(will be inserted by the editor)

New formulations for Variable Cost and Size Bin Packing Problems with Item Fragmentation

Marco Casazza

the date of receipt and acceptance should be inserted later

Abstract In the Bin Packing Problem with Item Fragmentation a set of items of known weight and a set of bins of limited capacity are given; the task is to find the minimum cost assignment of items to bins without exceeding their capacity. However, contrary to the classical Bin Packing Problem, items can be split and fractionally assigned to different bins at a cost. In this paper we generalize models and properties from the literature by considering a set of heterogeneous bins, possibly having different cost and capacity. We prove that such a natural extension changes substantial features of the problem. We propose both compact and extended formulations and a branch-and-price algorithm that combines column generation techniques and implicit enumeration strategies to achieve guarantees on the optimality of the solutions. We present the results of an extensive experimental campaign proving that our algorithm outperforms general purpose commercial solvers by orders of magnitude.

Keywords Bin Packing · Item Fragmentation · Variable Cost and Size · Column Generation

1 Introduction

In the classic *Bin Packing Problem (BPP)* a set of items of known weight and a set of bins of limited capacity are given. The task is to find an assignment of items to bins such that the sum of the items assigned to the same bin does not exceed its capacity. Thanks to their direct applicability to several logistic problems, BPPs are one of the most studied and yet lively category of decisional problems in combinatorial optimization. Furthermore, BPPs have also been shown to be suited for modeling network design problems, like message transmission in community TV networks [10]

Project partially funded by Regione Lombardia, POR FESR 2014-2020 project AD-COM

M. Casazza
Dipartimento di Informatica, Università degli Studi di Milano, via Bramante 65, 26013 Crema, Italy
E-mail: marco.casazza@unimi.it

and fully optical network planning problems [14]. In fact, it is easy to map network packets to items and communication channels to bins. However, due to the nature of such problems, where packets can be split among several channels, there is the need of additional features to BPP models.

The *BPP with Item Fragmentation (BPPIF)* has been introduced to model such an additional level of details: unlike BPPs it allows items to be split among several bins at a cost, generalizing the classic BPP and hence increasing its practical applicability. Several variants of BPPIFs have been discussed in the literature and for a full taxonomy we refer to [4]. For what concerns the way the capacity is consumed when items are split, the BPPIF can be found in two versions: *BPPIF with Size Increasing (BPPSIF)* and *BPPIF with Size Preserving (BPPSPF)*. In the former, each time an item is split an additional overhead is triggered, increasing its weight. In the latter, instead, the item weights remain constant. With regard to the objective function, two variants have been proposed in the literature, one called *fragmentation-minimization (fm-BPPIF)*, where a maximum number of bins is given and the number of fragmentations is minimized, and a second called *bin-minimization (bm-BPPIF)*, where an upper bound on the fragmentations is given and the number of bins is minimized.

BPPIF was first introduced in [10], where a proof of its NP-hardness was provided. In [12] and [13] traditional BPP heuristics are adapted to BPPIFs and their approximation properties are discussed. Fast and dual asymptotic fully polynomial time approximation schemes were presented in [16] and [15], while new approximation algorithms have been proposed in [9].

Indeed, we previously tackled BPPIFs in [2] and [4]: we highlighted theoretical properties on the structure of the optimal solutions, proposed new mathematical programming formulations, and a unified framework to solve several BPPIFs variants with optimality guarantees. In particular, in [4] we showed that both generic branch-and-cut and ad-hoc branch-and-price approaches benefit from formulations where the decision on the fractional assignment of an item to a bin is postponed at the end of the optimization process.

However, our exact methodologies have been tailored to the case where bins are homogeneous, which means that all bins are required to have the same cost and capacity. To achieve a higher degree of flexibility, BPPs find their natural generalization in the *Variable Cost and Size BPP* [7], where bins may differ in both cost and capacity. Therefore in this paper we address the heterogeneous variant of bm-BPPIFs, that we call *Variable Cost and Size BPPIF (VCSBF)*, proving that such an extension changes substantial features of our problem.

Our main contributions are the following:

– we investigate the complexity of BPPIFs when the number of allowed fragmentations is unbounded proving that, in such a case, the VCSBF is harder than the BPPIF with homogeneous bins [4];
– we prove that, similarly to the homogeneous case [4], we can model the VCSBF using a particular structure called *chain* that encodes a combinatorial number of ways to fragment items;
– we apply Dantzig-Wolfe decomposition to obtain an extended formulation and provide a column generation procedure to solve its continuous relaxation. In par-

ticular, we prove that although the pricing problem is different from the one in [4], it still can be solved in pseudo-polynomial time by decomposing it into two distinct and independent subproblems;
- we implement a branch-and-price algorithm including our column generation procedure to obtain tight dual bounds, valid dual cuts and a heuristic pricing procedure to speed up the bounding process, and ad-hoc branching rules to ensure the integrality of solutions;
- we prove that our methodology is solid and more effective than state-of-the-art methodologies through an extensive experimental campaign.

Preliminary partial results of our research were presented in [3] where we showed that a column generation procedure was able to provide tight dual bounds on a dataset adapted from [4]. In this manuscript we extend our theoretical study and provide proofs to our claims, we enforce our formulation and also improve the performance of our bounding procedure, we implement a complete branch-and-price algorithm and we test it against a new ad-hoc dataset.

For the ease of exposition, in Section 2 we formalize our optimization problem, extend a few theoretical properties, and propose its mathematical programming models. In Section 3 we show how we use decomposition techniques to obtain an extended formulation, while in Section 4 we detail our exact algorithm. We provide the results of our experimental campaign in Section 5 and we collect some brief conclusions in Section 6.

2 Modeling

We are given the set of items I and the set of bins B. Bins are heterogeneous: we are given the set T of all bin types, and B can be partitioned accordingly into disjointed sets B_t, one for each type $t \in T$. For each item $i \in I$ we are given its weight w_i, while for each bin of type $t \in T$ we are given both its capacity q_t and its cost c_t.

All items must be packed into bins. Split is allowed, and each item may be fragmented and fractionally assigned to different bins. The sum of the weights of the (fragments of) items packed into a single bin of type t must not exceed its capacity q_t. We say that a bin is open if it has at least one assigned (fragment of) item, and whenever a bin of type t is open we pay a cost c_t.

An upper bound $\mathscr{F}$ to the number of allowed fragmentations is given. The *Variable cost and size Bin Packing Problem with Item Fragmentation (VCSBF)* is the problem of finding a minimum cost packing containing all the items while not exceeding neither the number of fragmentations $\mathscr{F}$ nor the bin capacities. An example of a feasible packing is pictured in Figure 1.

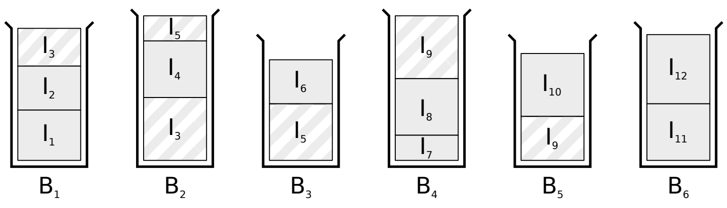

Fig. 1 Example of a feasible packing having 12 items (from I_1 to I_{12}) and using 6 bins (from B_1 to B_6). Items I_3, I_5, and I_9 are split and the packing has 3 fragmentations.

2.1 Compact formulation

We can describe the VCSBF by means of the following mathematical programming model:

$$\min \quad \sum_{t\in T} c_t \cdot \sum_{j\in B_t} u_j \tag{1}$$

$$\text{s.t.} \quad \sum_{t\in T}\sum_{j\in B_t} x_{ij} = 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{i\in I} w_i \cdot x_{ij} \le q_t \cdot u_j \qquad \forall t \in T, j \in B_t \tag{3}$$

$$\sum_{i\in I}\sum_{t\in T}\sum_{j\in B_t} z_{ij} - |I| \le \mathscr{F} \tag{4}$$

$$x_{ij} \le z_{ij} \qquad \forall i \in I, t \in T, j \in B_t \tag{5}$$

$$0 \le x_{ij} \le 1 \qquad \forall i \in I, t \in T, j \in B_t \tag{6}$$

$$z_{ij} \in \mathbb{B} \qquad \forall i \in I, t \in T, j \in B_t \tag{7}$$

$$u_j \in \mathbb{B} \qquad \forall t \in T, j \in B_t \tag{8}$$

where each variable x_{ij} is the fraction of item i packed into bin j, variable z_{ij} is set to 1 if a fraction of item i is packed into bin j and 0 otherwise, and variables u_j is set to 1 if bin j is open and 0 otherwise.

The objective function (1) minimizes the cost of the open bins. Constraints (2) impose that all items are fully assigned, while constraints (3) ensure that bins capacity is never exceeded. Constraint (4) avoid exceeding the allowed number of fragmentations. Constraints (5) ensure consistency between x_{ij} and z_{ij} variables.

Computational complexity. Both the VCSBF and its homogeneous version, the bm-BPPSPF, are $\mathbb{NP}$-hard in the general case: the latter was proven to be $\mathbb{NP}$-hard in [10], and it is easy to prove that also the former is at least as hard as the latter, since it is its generalization. However, when the upper bound on the number of fragmentation $\mathscr{F} = +\infty$, their computational complexity is different. In fact, if we consider an instance of bm-BPPSPF allowing infinite fragmentations, there always exists an optimal solution that can be found in polynomial time using the *Next-Fit with Item Fragmentation (NF$_f$)* algorithm proposed in [12]. Such an algorithm iteratively packs items into an open bin until the latter is full. When an item cannot be fully packed, it is split and fragmented among several bins. The current open bin is filled with a fraction of the item and then closed. Then a new bin is opened and the algorithm resumes the packing procedure starting from the remaining fraction of the fragmented item. When bins are homogeneous and $\mathscr{F} = +\infty$, such an algorithm always provides an optimal solution, since it never wastes any unit of capacity and the cost of a bin is paid only when necessary.

However, this is not the case for the VCSBF since bins are not equal ones. In fact we can prove that:

Theorem 1 *The VCSBF is $\mathbb{NP}$-hard even if the number of allowed fragmentations $\mathscr{F} = +\infty$.*

Proof When $\mathscr{F} = +\infty$, VCSBF can be reduced to:

$$\min \quad \sum_{t\in T} c_t \cdot \sum_{j\in B_t} u_j \tag{9}$$

$$\text{s.t.} \quad \sum_{t\in T}\sum_{j\in B_t} q_t \cdot u_j \geq \sum_{i\in I} w_i \tag{10}$$

$$u_j \in \mathbb{B} \qquad \forall t \in T, j \in B_t \tag{11}$$

where u_j is 1 if bin j is used and 0 otherwise. Such a problem, that we call VCSBF$_\infty$, is the problem of selecting the minimum cost subset of bins with at least enough capacity to pack all the items in I with an unbounded number of fragmentations.

Now, let us consider an instance of 0-1 *Knapsack Problem (KP)* such that we are given a knapsack of capacity q^{KP} and a set of items I^{KP}. For each item $i \in I^{KP}$ we are also given both its weight w_i^{KP} and its profit p_i^{KP}. Such an instance can be reduced to a VCSBF$_\infty$ instance as follows: each item $i \in I^{KP}$ corresponds to a bin type t having $q_t = w_i^{KP}$ and cost $c_t = p_i^{KP}$. Therefore for each type t we have $|B_t| = 1$ and $|B| = |T| = |I^{KP}|$. A single item $\hat{i} \in I$ is given having weight $w_{\hat{i}} = \sum_{i\in I^{KP}} w_i^{KP} - q^{KP}$.

By solving such an instance of VCSBF$_\infty$ we also solve KP: the subset of bins selected in a VCSBF$_\infty$ solution corresponds to a subset of knapsack items having minimum profit, and therefore the set of non-selected bins corresponds to a set of knapsack items having maximum profit. Furthermore, the set of non-selected knapsack items does not exceed the knapsack capacity q^{KP}. A solution to KP can then be obtained by solving VCSBF$_\infty$ and then packing all the knapsack items corresponding to bins having $u_j = 0$.

Since an instance of KP can be reduced to an instance of VCSBF$_\infty$ in polynomial time and KP is $\mathbb{NP}$-hard, VCSBF$_\infty$ is $\mathbb{NP}$-hard. □

2.2 Chain formulation

Even though model (1) – (8) can be used to solve VCSBF instances, from a theoretical point of view it is easy to prove that the dual bound provided by its continuous relaxation is trivial: in fact, an optimal solution to its continuous relaxation can be found by iteratively selecting bins having minimum c_t/q_t until $\sum_{i\in I} w_i$ units of capacity are collected. Feasible x_{ij} and z_{ij} values can be found by simply applying NF$_\text{f}$ algorithm assuming infinite fragmentations, while variables u_j can be set to $u_j = \sum_{i\in I} w_i \cdot x_{ij}/q_t$ for all $t \in T, j \in B_t$, obtaining an optimal solution to the continuous relaxation.

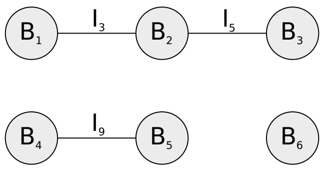

Fig. 2 BPG representation of solution in Figure 1: each vertex corresponds to a bin in the solution, while each edge connect vertices sharing a fragmented item. The BPG is composed by three connected components: $\{B_1,B_2,B_3\}$, $\{B_4,B_5\}$, and $\{B_6\}$.

Furthermore, in both [2] and [4] it has been proved that from an empirical point of view generic cutting planes approaches applied to this compact formulation provide poor dual bounds. Instead, by removing the decision variables on the fractional assignments of items to bins, new and more effective formulations can be designed. We now follow this principle by proving that our problem allows such type of formulations.

We start by recurring to the definitions of *Bin Packing Graph (BPG)* and *primitive solution*, both introduced in [15]: the former is a graph having a vertex for each bin and an edge for each pair of vertices sharing an item (see Figure 2), while the latter is a feasible solution to BPPIF such that (a) each bin contains at most two fragmented items, (b) each item is fragmented at most once, and (c) the corresponding BPG is a collection of paths. Still in [15] it has been proved that:

Theorem 2 ([15]) *Any instance of a bm-BPPIF has an optimal solution which is primitive.*

It is easy to extend Theorem 2 to the case with heterogeneous bins and prove that:

Theorem 3 *Given an instance of VCSBF having $w_i \leq q_j$, for each pair of item i and bin j, there always exists an optimal solution that is primitive.*

Proof Following the proof in [15] and [4], we observe that if we are given the BPG of an optimal solution to VCSBF, each of its connected components can be transformed into a path: all the items assigned to the bins of a connected component can be packed into the same bins using the $\mathrm{NF_f}$ algorithm. The BPG obtained after repacking all the items of each connected component corresponds to a collection of paths in which each item is fragmented at most once and each bin has at most two fragmented items. □

In [4], we exploited such a property to propose what we called a *chain* formulation: given the BPG of a primitive solution, we say that a chain is the set of bins corresponding to a BPG path, and it is characterized by a cost and a capacity that are the sum of the costs and the sum of the capacities of its bins, respectively. An example of chain representation of the solution in Figure 1 is depicted in Figure 3. We also show that our problem can be reduced to the problem of packing items into a minimum cost set of chains, instead of a minimum cost set of bins. This is true also for VCSBF, since it allows optimal primitive solutions when $w_i \leq q_j$. However, we can further extend such a property:

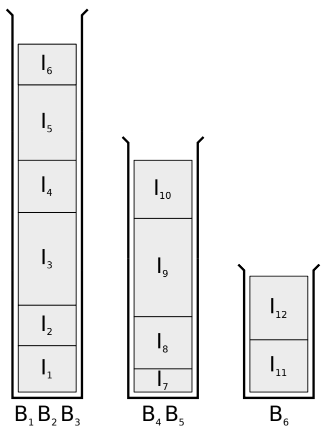

Fig. 3 Chain representation of solution in Figure 1: bins belonging to the same connected component in the BPG are stacked in order to form a chain of bins. The capacity of each chain is the sum of the capacities of its bins.

Theorem 4 *Any instance of VCSBF can be modeled with a chain formulation.*

Proof Given a connected component of the BPG of an optimal solution to VCSBF, we can always obtain a cost equivalent packing by simply applying NF_f algorithm on the same set of bins corresponding to the connected component. If there exists an item i having $w_i > q_j$, then we may split it more than once among more than 2 bins, therefore obtaining a solution that is not primitive. However, the solution obtained would still use the same bins and perform at most the same number of fragmentations, since the number of fragmentations obtained through NF_f is always less than the number of bins used for the packing, as proven in [12]. Therefore, given any connected component of the BPG, we can apply NF_f to obtain a cost equivalent chain that does not exceed neither the number of bins nor the number of fragmentations.

□

Let K be the set of chains; w.l.o.g. $|K| = |B|$, since each chain corresponds to a set of at least one bin, and therefore no solution using more than $|B|$ chains can be feasible.

Model(1) – (8) can be then formulated as:

$$\min \sum_{k\in K}\sum_{t\in T} c_t \cdot v_{tk} \tag{12}$$

$$\text{s.t.} \sum_{k\in K} z_{ik} = 1 \qquad \forall i \in I \tag{13}$$

$$\sum_{i\in I} w_i \cdot z_{ik} \leq \sum_{t\in T} q_t \cdot v_{tk} \qquad \forall k \in K \tag{14}$$

$$\sum_{k\in K} (\sum_{t\in T} v_{tk}) - u_k \leq \mathscr{F} \tag{15}$$

$$u_k \leq \sum_{t\in T} v_{tk} \qquad \forall k \in K \tag{16}$$

$$\sum_{k\in K} v_{tk} \leq |B_t| \qquad \forall t \in T \tag{17}$$

$$z_{ik} \in \mathbb{B} \qquad \forall i \in I, k \in K \tag{18}$$

$$u_k \in \mathbb{B} \qquad \forall k \in K \tag{19}$$

$$v_{tk} \in \mathbb{N}_0 \qquad \forall t \in T, k \in K \tag{20}$$

where each variable z_{ik} is 1 if item i is assigned to chain k and 0 otherwise, each variable u_k is 1 if chain k is used and 0 otherwise, and each variable v_{tk} is the number of bins of type t in chain k.

The objective function (12) minimizes the cost of the packing. Constraints (13) ensure that all items are packed, while constraints (14) impose that the capacities of the chains are not exceeded. Constraint (15) imposes a bound to the maximum number of fragmentations. Constraints (16) ensure that each selected chain has at least one bin. Constraints (17) ensure that we do not exceed the number of bins of each type.

From a continuous perspective, our chain formulation achieves the same trivial dual bound of the compact formulation. In fact, a feasible solution to the continuous relaxation of the compact model is also feasible to the continuous relaxation of the chain formulation.

3 Extended formulation

In order to achieve an improved dual bound, we now propose a reformulation of our problem obtained through Dantzig-Wolfe decomposition [8]. Let $\mathrm{z}_k = (z_{1k}, \ldots, z_{|I|k})$, $\mathrm{v}_k = (v_{1k}, \ldots, v_{|T|k})$, $\mathrm{w} = (w_1, \ldots, w_{|I|})$, and $\mathrm{q} = (q_1, \ldots, q_{|T|})$. For each $k \in K$ let

$$\Omega_k = \left\{ (\mathrm{z}_k, u_k, \mathrm{v}_k) \in \mathbb{B}^{|I|+1+|T|} \mid \mathrm{w}^T \cdot \mathrm{z}_k \leq \mathrm{q}^T \cdot \mathrm{v}_k \wedge u_k \leq ||\mathrm{v}_k||_1 \right\}$$

be the set of feasible integer points with respect to constraints (14), (16), and (18) – (20). We relax the integrality conditions but replace each Ω_k with the convex hull of its L_k extreme integer points

$$\Gamma_k = \left\{ (\bar{\mathrm{z}}_k^1, \bar{u}_k^1, \bar{\mathrm{v}}_k^1), \ldots, (\bar{\mathrm{z}}_k^{L_k}, \bar{u}_k^{L_k}, \bar{\mathrm{v}}_k^{L_k}) \right\}$$

and we impose

$$(\mathrm{z}_k, u_k, \mathrm{v}_k) = \sum_{l \in \Gamma_k} (\bar{\mathrm{z}}_k^l, \bar{u}_k^l, \bar{\mathrm{v}}_k^l) \cdot y_k^l, \tag{21}$$

where $y_k^l \geq 0$ for each $k \in K$ and $l \in \Gamma_k$, and $\sum_{l \in \Gamma_k} y_k^l = 1$ for each $k \in K$.

The model obtained by replacing in the continuous relaxation of formulation (12) – (20) the vectors $(\mathrm{z}_k, u_k, \mathrm{v}_k)$ as indicated in (21), and by making explicit the indexes is:

$$\min \sum_{k \in K} \sum_{l \in \Gamma_k} \sum_{t \in T} c_t \cdot \bar{v}_{tk}^l \cdot y_k^l \tag{22}$$

$$\text{s.t.} \sum_{k \in K} \sum_{l \in \Gamma_k} \bar{z}_{ik}^l \cdot y_k^l = 1 \qquad \forall i \in I \tag{23}$$

$$\sum_{k \in K} \sum_{l \in \Gamma_k} \bar{v}_{tk}^l \cdot y_k^l \leq |B_t| \qquad \forall t \in T \tag{24}$$

$$\sum_{k \in K} \sum_{l \in \Gamma_k} ((\sum_{t \in T} \bar{v}_{tk}^l) - \bar{u}_k^l) \cdot y_k^l \leq \mathscr{F} \tag{25}$$

$$\sum_{l \in \Gamma_k} y_k^l = 1 \qquad \forall k \in K \tag{26}$$

$$y_k^l \geq 0 \qquad \forall k \in K, \forall l \in \Gamma_k \tag{27}$$

Constraints (23) can be relaxed in $\geq$ form, since an optimal solution always exists in which no item is assigned more than once. Constraints (26) can be relaxed in $\leq$ form because an empty pattern with $\bar{\mathrm{z}}_k^l = 0$, $\bar{u}_k^l = 0$, and $\bar{\mathrm{v}}_k^l = 0$ always exists for each $k \in K$. We can also relax constraint (25) in

$$\sum_{k \in K} \sum_{l \in \Gamma_k} ((\sum_{j \in B} \bar{v}_{jk}^l) - 1) \cdot y_k^l \leq \mathscr{F}$$

because $u_k^l = 1$ for all patterns but the empty ones.

We observe that all sets Γ_k are identical, since identical are the sets of bins composing the chains. Therefore, we consider a single representative $\Gamma = \bigcap_{k \in K} \Gamma_k$, and aggregate constraints (26) as

$$\sum_{l \in \Gamma} y^l \leq |K|$$

which can then be removed since there always exists an optimal solution where at most $|K|$ patterns are selected, since each pattern l is composed at least by a bin and $|K| = |B|$.

We then obtain the following Master Problem (MP):

$$\min \sum_{l\in\Gamma}\sum_{t\in T} c_j \cdot \vec{v}^l_t \cdot y^l \tag{28}$$

$$\text{s.t.} \sum_{l\in\Gamma} \vec{z}^l_i \cdot y^l \geq 1 \qquad \forall i \in I \tag{29}$$

$$\sum_{l\in\Gamma} \vec{v}^l_t \cdot y^l \leq |B_t| \qquad \forall t \in T \tag{30}$$

$$\sum_{l\in\Gamma} ((\sum_{t\in T} \vec{v}^l_j) - 1) \cdot y^l \leq \mathscr{F} \tag{31}$$

$$y^l \geq 0 \qquad \forall l \in \Gamma \tag{32}$$

Observation 1 *The dual bound provided by the MP is at least as good as the one given by the continuous relaxation of chain formulation.*

Such observation directly follows from the Dantzig-Wolfe decomposition principle.

4 Algorithm

We remark that formulation (28) – (32) has a number of variables that is exponential in both the number of items and bins; therefore, we recur to *Column Generation (CG)* techniques: we start with a *Restricted Master Problem (RMP)* having a small set of variables only, and solve it to optimality. We collect dual information and then solve a combinatorial problem to find negative reduced cost variables (see Subsection 4.2). If any is found, we add it to the RMP and the CG process is repeated, otherwise both the RMP and the MP are optimal, and their value is a valid lower bound for the VCSBF. If the final RMP solution is also integer, then it is optimal for VCSBF, otherwise we enter a search tree to find a proven global optimum.

4.1 Initialization

The initialization of the RMP allows to start the CG process having already a small set of variables and may reduce the heading-in effect [17]. In our algorithm we found to be profitable to initialize our RMP with two sets of columns: the first is the set of columns having one single item packed into the minimum cost bin that can contain it; in such a way, when items are smaller than at least one bin, we always start the CG process having a trivial but yet feasible solution. The second set of columns is the set of dual cuts already proposed in [4], and it is based on the observation that given the non-negative dual variables λ_i corresponding to constraints (29) the following holds:

Proposition 1 **([4])** *Given two subsets $S \subseteq I$ and $T \subseteq I$ such that $S \cap T = \emptyset$ and $\sum_{i\in S} w_i \leq \sum_{i\in T} w_i$, no optimal dual solution packing all items in T violates the inequality $\sum_{i\in S} \lambda_i \leq \sum_{i\in T} \lambda_i$.*

In fact, all columns representing chains including all items in T can be replaced by columns where items in T are removed and replaced by items in S, obtaining new columns that are both feasible and improving.

For what concerns the implementation, for each dual cut we include in the RMP a column having $\bar{z}_i^l = 1$ for each $i \in S$ and $\bar{z}_i^l = 0$ for each $i \in T$. Since these cuts are in exponential number, we found to be already profitable to include the subsets having $|S| = 1$ and $|T| = 1$ only.

4.2 Pricing new variables

Let $\lambda \geq 0$, $\mu \leq 0$, and $\eta \leq 0$ be respectively the dual variables of constraints (29), (30), and (31). The pricing problem can be described as:

$$\sigma^* = \min \sum_{t\in T} c_t \cdot v_t - \sum_{i\in I} \lambda_i \cdot z_i - \sum_{t\in T} \mu_t \cdot v_t - \eta \cdot ((\sum_{t\in T} v_t) - 1) \tag{33}$$

$$\text{s.t.} \sum_{i\in I} w_i \cdot z_i \leq \sum_{t\in T} q_t \cdot v_t \tag{34}$$

$$\sum_{t\in T} v_t \leq \mathscr{F} + 1 \tag{35}$$

$$z_i \in \mathbb{B} \qquad \forall i \in I \tag{36}$$

$$v_t \in \mathbb{N}_0 \qquad \forall t \in T \tag{37}$$

The objective function (33) can be rewritten in maximization form as:

$$\sigma^* = -\max \sum_{i\in I} \lambda_i \cdot z_i - \sum_{t\in T} (c_t - \mu_t - \eta) \cdot v_t - \eta \tag{38}$$

where $\lambda_i \geq 0$ is the profit collected when item i is packed into a chain, $(c_t - \mu_t - \eta) \geq 0$ is the cost paid for each bin of type t in the chain, and $\eta \leq 0$ is a fixed profit. We call problem (33) – (37) *Variable Cost and Size* 0-1 *Knapsack Problem (VCSKP)*.

Intuitively, we can observe that the objective function is composed by two different part: the first part promotes solutions packing more items, and therefore consuming more capacity, while the second part, instead, puts a cost on each unit of capacity consumed, therefore pushing in the opposite direction. In order to design an effective algorithm to solve the VCSKP we can first observe that:

Observation 2 *Given $\mathscr{Q} \geq 0$, an optimal solution to VCSKP using at most $\mathscr{Q}$ units of capacity can be found by solving two distinct and independent subproblems $(SP1)$ and $(SP2)$, one maximizing the profit and one minimizing the cost, respectively.*

The two subproblems can be modeled as follows:

$$(SP1) \begin{cases} \sigma^*_{sp1}(\mathscr{Q}) = \max \sum_{i\in I} \lambda_i \cdot z_i & (39) \\ \text{s.t.} \sum_{i\in I} w_i \cdot z_i \leq \mathscr{Q} & (40) \\ z_i \in \mathbb{B} \qquad \forall i \in I & (41) \end{cases}$$

$$
(SP2)\begin{cases}
\sigma^*_{sp2}(\mathscr{Q}) = \min \sum_{t\in T} (c_t - \mu_t - \eta)\cdot v_t & (42)\\
\text{s.t.} \sum_{t\in T} q_t \cdot v_t \geq \mathscr{Q} & (43)\\
\sum_{t\in T} v_t \leq \mathscr{F} + 1 & (44)\\
v_t \in \mathbb{N}_0 \qquad \forall t \in T & (45)
\end{cases}
$$

$(SP1)$ maximizes the profit of a knapsack consuming at most $\mathscr{Q}$ units of capacity, while $(SP2)$ select the minimum cost subset of bins providing $\mathscr{Q}$ units of capacity. The knapsack obtained by packing the items of a solution to $(SP1)$ into a chain of bins of a solution to $(SP2)$ is feasible for VCSKP. However, even if the value $\sigma^*_{sp1}(\mathscr{Q}) - \sigma^*_{sp2}(\mathscr{Q})$ is optimal for a knapsack using at most $\mathscr{Q}$ units of capacity, it may not be optimal for VCSKP.

Observation 3 *Given $\bar{Q} = \min(\sum_{t\in T} q_t \cdot |B_t|, \sum_{i\in I} w_i)$, an optimal solution to VCSKP can be found by solving subproblems $(SP1)$ and $(SP2)$ for each value of $\mathscr{Q}$, that is*

$$
\sigma^* = -\max_{0\leq \mathscr{Q} \leq \bar{Q}} (\sigma^*_{sp1}(\mathscr{Q}) - \sigma^*_{sp2}(\mathscr{Q}) - \eta) \tag{46}
$$

In fact, among all values of $\mathscr{Q}$ there must be at least one corresponding to an optimal knapsack.

Lemma 1 $(SP1)$ *can be solved in pseudo-polynomial time.*

Proof $(SP1)$ is the classic 0-1 KP, which can be solved in pseudo-polynomial time by means of the well-known dynamic programming algorithm [11]: let $M_{sp1}(\bar{I}, q)$ be the value of an optimal solution to $(SP1)$ in which only items $\bar{I} \subseteq I$ can be selected, and q units of capacity are consumed. $M_{sp1}(\bar{I}, q)$ can be computed recursively:

$$
M_{sp1}(\bar{I} \cup \{i\}, q) = \begin{cases} M_{sp1}(\bar{I}, q), \text{ if } w_i > q \\ \max\{M_{sp1}(\bar{I}, q), M_{sp1}(\bar{I}, q - w_i) + \lambda_i\}, \text{ otherwise} \end{cases}
$$

where $M_{sp1}(\emptyset, q) = 0$ for each $0 \leq q \leq \mathscr{Q}$. Thus, $\sigma^*_{sp1}(\mathscr{Q}) = \max_{0\leq q\leq \mathscr{Q}} M_{sp1}(I, q)$, which can be computed in $O(|I| \cdot \mathscr{Q})$ time. □

Lemma 2 $(SP2)$ *can be solved in pseudo-polynomial time.*

Proof Using an auxiliary variable $\bar{v}_t = |B_t| - v_t$ indicating how many bins of type t are excluded from the solution, we can reformulate $(SP2)$ as:

$$
(\overline{SP2})\begin{cases}
\sigma^*_{\overline{sp2}}(\mathscr{Q}) = \max \sum_{t\in T} (c_t - \mu_t - \eta)\cdot \bar{v}_t & (47)\\
\text{s.t.} \sum_{t\in T} q_t \cdot \bar{v}_t \leq \sum_{t\in T} q_t \cdot |B_t| - \mathscr{Q} & (48)\\
\sum_{t\in T} \bar{v}_t \geq |B| - (\mathscr{F} + 1) & (49)\\
\bar{v}_t \in \mathbb{N}_0 \qquad \forall t \in T & (50)
\end{cases}
$$

with $\sigma^*_{sp2} = \sum_{t \in T}(c_t - \mu_t - \eta) \cdot |B_t| - \sigma^*_{\overline{sp2}}$.

$(\overline{SP2})$ can be solved in a similar way to $(SP1)$: let $\bar{\mathscr{Q}} = \sum_{t \in T} q_t \cdot |B_t| - \mathscr{Q}$ and let $M_{\overline{sp2}}(\bar{B}, f, q)$ be the cost of an optimal solution to $(\overline{SP2})$ where at least f bins in $\bar{B} \subseteq B$ are selected, and at most q units of capacity are consumed. $M_{\overline{sp2}}(\bar{B}, f, q)$ can be computed recursively:

$$M_{\overline{sp2}}(\bar{B} \cup \{j\}, f, q) \\ = \begin{cases} M_{\overline{sp2}}(\bar{B}, f, q), \text{ if } q_j > q \\ \max\{M_{\overline{sp2}}(\bar{B}, f, q), M_{\overline{sp2}}(\bar{B}, f-1, q-q_j) + (c_j - \mu_j - \eta)\}, \text{ otherwise} \end{cases}$$

where $M_{\overline{sp2}}(\emptyset, f, 0) = 0$ and $M_{\overline{sp2}}(\emptyset, 0, q) = 0$ for each $0 \leq f \leq |B|$ and $0 < q \leq \bar{\mathscr{Q}}$. Thus,

$$\sigma^*_{\overline{sp2}}(\mathscr{Q}) = \max_{0 \leq f \leq \mathscr{F}+1, 0 \leq q \leq \bar{\mathscr{Q}}} M_{\overline{sp2}}(B, |B| - f, q). \tag{51}$$

Similarly to the classic 0-1 KP, the problem can be solved by means of a three-dimensional matrix of size $|B| \cdot |B| \cdot \bar{\mathscr{Q}}$. However, in an optimal solution to $(\overline{SP2})$, we never select more than $\mathscr{F}+1$ bins of each type, and therefore, for each type t we have that $v_t \leq \mathscr{F}+1$ and $\bar{v}_t \geq |B_t| - (\mathscr{F}+1)$. Therefore, in our algorithm we can consider the last $\mathscr{F}+1$ bins of each type and reduce the size of the matrix we use to solve the problem to $|T| \cdot (\mathscr{F}+1) \cdot (\mathscr{F}+1) \cdot \bar{\mathscr{Q}}$. $(\overline{SP2})$ can be then solved in $O(|T| \cdot \mathscr{F}^2 \cdot \bar{\mathscr{Q}})$ time. □

Theorem 5 *An optimal solution to the VCSKP exists that can be found by a pseudo-polynomial time algorithm.*

Proof The proof follows directly from Observation 3 and Lemmas 1 and 2: VCSKP can be decomposed in $\bar{Q}$ subproblems that can be solved in pseudo-polynomial time. However, we remark that by computing once $M_{sp1}(I, \bar{Q})$ and $M_{sp2}(B, \mathscr{F}, \bar{Q})$ we solve $(SP1)$ and $(SP2)$ for all $0 \leq \mathscr{Q} \leq \bar{Q}$. Therefore, to solve VCSKP only two subproblems $(SP1)$ and $(SP2)$ must be solved, and our procedure is a pseudo-polynomial time algorithm. □

Heuristic pricing. In a preliminary experimental analysis we found to be profitable to speed up our CG procedure by adding a heuristic variant of our pricing algorithm. In fact, we can observe that when constraint (49) is removed from $(\overline{SP2})$, the problem is exactly a 0-1 KP, which can be solved faster than the actual $(\overline{SP2})$. However, if constraint (49) is removed we may generate columns corresponding to chains of more than $\mathscr{F}+1$ bins, and thus potentially reducing the quality of the dual bound. We then solve our pricing problem setting $\bar{Q} = \min((\mathscr{F}+1) \cdot \min_{t \in T} q_t, \sum_{i \in I} w_i)$, in such a way we always generate columns having at most $\mathscr{F}+1$ bins.

4.3 Branch and bound

When the optimal MP is fractional we explore a search tree by means of branching operations. In the following we propose three branching rules that first ensure the integrality of the number of bins of each type selected in a solution, and then fix the assignment of both items and bins to chains. While the latter are the only ones needed to ensure the integrality of the solutions, we observed that the former speeds up the pruning of non-optimal branches.

Fixing the number of bins for each type. Let $\tilde{y}^l$ be the value of a variable y^l in a fractional solution of the MP. We start by observing that if all $\tilde{y}^l$ are integer, then an integer solution is found, we eventually update the primal bound, and the branching node is pruned. Otherwise, let $\tilde{v}_t = \sum_{l \in \Gamma} \bar{v}^l_t \cdot \tilde{y}^l$ be the fractional number of bins of type t selected in a fractional solution to MP. If all $\tilde{v}_t$ values are integer, than the solution is integer with respect with the current branching rule and we skip to the next one. Otherwise, we search for the bin type $\hat{t}$ having the most fractional number of bins in the fractional solution, that is $\hat{t} \in \operatorname{argmin}_{t \in T} |\tilde{v}_t - (\lfloor \tilde{v}_t \rfloor + 0.5)|$, and we create two children: in the first we impose a maximum number of $\lfloor \tilde{v}_{\hat{t}} \rfloor$ bins of type $\hat{t}$ by adding a new constraint $\sum_{l \in \Gamma} \bar{v}^l_{\hat{t}} \cdot y^l \leq \lfloor \tilde{v}_{\hat{t}} \rfloor$ in the MP; vice versa, in the second child we impose a minimum number $\lceil \tilde{v}_{\hat{t}} \rceil$ by adding a new constraint $\sum_{l \in \Gamma} \bar{v}^l_{\hat{t}} \cdot y^l \geq \lceil \tilde{v}_{\hat{t}} \rceil$. In both cases a new dual variable must be taken into account while solving the MP, but neither the structure of the pricing problem nor its implementation are affected.

Fixing items into chains. When the number of bin types is integer, we look for the fractional assignments of items to chains similarly as we did in [4]: the idea is that for each chain we can select one of its packed items to be the one that identifies the entire chain. Such an item is then called *head* item. Let us suppose that each node of the branching tree holds a set $H \subseteq I$ of head items, each corresponding to a chain. We start with $H = \emptyset$, and during the exploration of the search tree, each item is either selected to be assigned to one of the chains identified by head items or it becomes a new head itself.

Such a branching is divided in two phases:

1. If $H = \emptyset$ we skip to Phase 2 to select a new head item. Otherwise, let $\tilde{z}_{hi} = \sum_{l \in \Gamma} \bar{z}_h \cdot \bar{z}_i \cdot \tilde{y}^l$ be the fractional value of the number of chains in which item i is selected with head h. We select the pair $(\hat{h}, \hat{\imath})$ having the most fractional $\tilde{z}_{hi}$, that is $(\hat{h}, \hat{\imath}) \in \operatorname{argmin}_{h \in H, i \in I \setminus H} |\tilde{z}_{hi} - 0.5|$, and create two children, one where we impose that item $\hat{\imath}$ is never packed with head $\hat{h}$, and one where i is always and only packed with $\hat{h}$. To impose such constraints, before processing a node we remove from the RMP all the columns that do not conform to the current branching decisions and solve $|H| + 1$ pricing problems, one for each chain, and one for the items that have not been assigned yet.
2. If no fractional assignment of items to chains is found, then it holds that no column having an head item is fractionally selected. However, this is not sufficient to ensure the integrality of the MP solution, because fractional assignments may arise in the set of columns having no fixed head item. Therefore, we search for the most fractional chain having no fixed head item, that is $\hat{l} \in \operatorname{argmin}_{l \in \Gamma | \bar{z}^l_h = 0, \forall h \in H} |\tilde{y}^l - 0.5|$. We select an arbitrary item $\hat{\imath}$ such that $\bar{z}^{\hat{l}}_{\hat{\imath}} = 1$, we extend the head items set $H \leftarrow H \cup \{\hat{\imath}\}$, and restart the branching from Phase 1.

Fixing bins into chains. Once all items are fixed into chains, we perform the same strategy with respect to the number of bin types selected in each chain. In fact, in a MP solution we may have chains that are composed by a fractional number of bin types. Let $\tilde{w}_{ht} = \sum_{l \in \Gamma} \bar{z}_h \cdot \bar{v}_t \cdot \bar{y}^l$ be the fractional number of bin of type t that are used in a chain identified by head item h. We search for the pair of $(\hat{h}, \hat{t})$ having the most fractional value $\tilde{w}_{ht}$, that is $(\hat{h}, \hat{t}) \in \operatorname{argmin}_{h \in H, t \in T} |\tilde{w}_{ht} - (\lfloor \tilde{w}_{ht} \rfloor + 0.5)|$. If all $\tilde{w}_{ht}$ values are integer, then an integer solution is found, we eventually update the primal bound and prune the node. Otherwise we create two children, one where the chain identified by head item $\hat{h}$ has at most $\lfloor \tilde{w}_{\hat{h}\hat{t}} \rfloor$ bins of type $\hat{t}$, and one where it has at least $\lceil \tilde{w}_{\hat{h}\hat{t}} \rceil$ bins of type $\hat{t}$.

5 Experimental analysis

We implemented our algorithms in C++, using the framework SCIP [1] version 4.0.0, setting the default options. The LP subproblems were solved using the simplex algorithm implemented in CPLEX 12.6.3 [6] setting the automatic selection between primal and dual methods. As primal heuristic we included a single LP rounding heuristic provided by SCIP after the evaluation of each node of the branching tree. In the remainder we refer to our exact branch-and-price algorithm as BPA.

As a benchmark we considered the branch-and-cut implemented in CPLEX version 12.6.3, using both the classic formulation described in Subsection 2.1 and the chain formulation in Subsection 2.2 with default settings. In the remaining, we refer to the first as BC and to the second as CHAINBC.

All the tests have been conducted on a PC equipped with an Intel(R) Core(TM) i7-6700K CPU at 4.00GHz with 32 GB of memory and forcing single thread execution for all the three competitors.

For our experiments we produced a random dataset of instances, created in a similar fashion to [5,7]: we generated 10 instances for each combination of

- number of items: $|I| \in \{10, 25, 50, 75, 100\}$;
- weights distribution: small ($w_i \in [1, 100]$), medium ($w_i \in [25, 100]$), and large ($w_i \in [50, 100]$);
- set of types of bins: set1 ($|T| = 2$ and $q_t \in \{100, 120\}$), set2 ($|T| = 3$ and $q_t \in \{80, 100, 120\}$), set3 ($|T| = 4$ and $q_t \in \{80, 100, 120, 140\}$), set4 ($|T| = 5$ and $q_t \in \{60, 80, 100, 120, 140\}$);
- number of fragmentations: $\mathscr{F} = \lfloor |I|/f \rfloor$, with $f \in \{3, 5, 10\}$.

For each type of bins t, the cost is computed as $c_t = \lfloor 100 \cdot \sqrt{q_t} \rfloor$.

5.1 Lower bounds

In a first round of experiments we compared the quality of the lower bound obtained by the three methodologies when stopped at the root node of the branching tree and the computational effort required to compute it.

In Table 1 we report the aggregated results for number of items, weight of items, and number of fragmentations; for all of the three methodologies we show the average gap $\overline{g} = (BN - LB)/BN$ between the lower bound LB and the best known solution value BN found with either one of the three methodologies after one hour of computation (see Subsection 5.2), the average gap $\underline{g} = (\overline{LB} - LB)/\overline{LB}$ between the lower bound LB and the best lower bound $\overline{LB}$ found by the three methodologies, and the computing time t.

Our results show that the lower bound obtained by BC is on average the worst and requires the highest computational effort to be computed. CHAINBC is on average the fastest, and its bounds are comparable with the ones obtained by our BPA. However, our BPA is the only methodology that always finds a lower bound that is at most 1% far from the best lower bound. Also, we can observe that our BPA outperforms the other methodologies when the number of allowed fragmentations is strict and weights are large: these are the case where item fragmentation is almost mandatory but the fragmentation resource is scarce.

Although the quality of the lower bounds obtained by both BC and CHAINBC on our dataset differs from the one on the BPPIF with homogeneous bins [4], we can still state that when fragmenting is profitable state-of-the-art methodologies provide less effective lower bounds.

5.2 Optimality guarantees

In our second round of experiments we ran a comparison between the three methodologies when solving our instances to proven optimality, setting a time limit of one hour for each run.

In Table 2 we report for each methodology the number of instances solved to proven optimality s, the average duality gap $g = (UB - LB)/UB$ between the lower bound LB and the upper bound UB when the time limit is hit, and the average computing time t for instances solved to proven optimality only.

The results show that our BPA outperforms both BC and CHAINBC approaches, solving to proved optimality 1798 instances out of 1800, while BC and CHAINBC solve 711 and 695 instances, respectively. As for the previous run of experiments, we can observe a common trend for both BC and CHAINBC results: classes of instances with smaller items or a high number of allowed fragmentation are easier to solve. In fact, having smaller items allows less fragmentation and instances are more related to classic BPPs. Also, when instances have a high number of allowed fragmentation it may be possible to find an optimal solution by simply solving a 0-1 KP, as shown in Theorem 1 for the asymptotic case with $\mathscr{F} = \infty$. Contrary to BC and CHAINBC, our BPA is faster on those instances where the fragmentation resource is scarce.

6 Conclusions

In this paper we addressed a variant of BPP where items can be split at a cost. We extended properties from the literature to a more generic case where a set of heterogeneous bins is given, proposing new mathematical programming models that avoid the

Table 1 Lower bound obtained at root node of the branching tree.

Instance class			BC			CHAINBC			BPA		
$\|I\|$	weight	$\mathscr{F}$	$\overline{g}(\%)$	$\underline{g}(\%)$	$t(s)$	$\overline{g}(\%)$	$\underline{g}(\%)$	$t(s)$	$\overline{g}(\%)$	$\underline{g}(\%)$	$t(s)$
10	small	1	2.4	1.7	0.1	2.0	1.4	0.0	0.9	0.1	0.0
10	small	2	1.4	0.5	0.1	1.2	0.3	0.0	1.2	0.3	0.0
10	small	3	1.0	0.3	0.1	0.8	0.1	0.0	0.9	0.3	0.0
10	medium	1	3.7	2.7	0.1	3.6	2.6	0.0	1.2	0.1	0.0
10	medium	2	0.8	0.5	0.1	0.7	0.4	0.0	0.5	0.1	0.0
10	medium	3	0.7	0.3	0.1	0.7	0.2	0.0	0.8	0.3	0.0
10	large	1	5.5	5.0	0.2	5.3	4.8	0.0	0.5	0.0	0.0
10	large	2	1.5	1.1	0.1	1.2	0.9	0.0	0.4	0.0	0.0
10	large	3	0.8	0.5	0.2	0.6	0.3	0.0	0.5	0.2	0.0
25	small	2	0.6	0.1	1.3	0.6	0.1	0.1	0.6	0.1	0.1
25	small	5	0.6	0.1	1.2	0.5	0.0	0.1	0.6	0.1	0.1
25	small	8	0.6	0.1	1.0	0.5	0.0	0.0	0.6	0.1	0.1
25	medium	2	1.3	0.9	1.3	1.3	0.8	0.1	0.5	0.0	0.1
25	medium	5	0.4	0.1	1.2	0.3	0.0	0.1	0.4	0.1	0.1
25	medium	8	0.4	0.1	1.0	0.3	0.0	0.1	0.4	0.1	0.1
25	large	2	6.7	6.2	1.8	6.7	6.2	0.1	0.6	0.0	0.0
25	large	5	1.2	0.6	1.4	1.2	0.6	0.1	0.7	0.0	0.1
25	large	8	0.3	0.0	1.2	0.3	0.0	0.1	0.3	0.1	0.1
50	small	5	0.2	0.0	9.4	0.2	0.0	0.2	0.2	0.0	0.4
50	small	10	0.2	0.0	7.8	0.2	0.0	0.2	0.2	0.0	0.4
50	small	16	0.2	0.0	7.6	0.2	0.0	0.2	0.2	0.0	0.4
50	medium	5	0.3	0.1	10.2	0.2	0.1	0.3	0.2	0.0	0.3
50	medium	10	0.2	0.0	8.9	0.2	0.0	0.2	0.2	0.0	0.4
50	medium	16	0.2	0.0	8.1	0.2	0.0	0.2	0.2	0.0	0.5
50	large	5	4.6	4.2	11.9	4.5	4.1	0.4	0.5	0.0	0.2
50	large	10	1.0	0.5	10.3	1.0	0.5	0.3	0.5	0.0	0.4
50	large	16	0.2	0.0	9.0	0.1	0.0	0.2	0.2	0.0	0.5
75	small	7	0.1	0.0	25.9	0.1	0.0	0.7	0.1	0.0	1.2
75	small	15	0.1	0.0	20.9	0.1	0.0	0.6	0.1	0.0	1.3
75	small	25	0.1	0.0	20.2	0.1	0.0	0.5	0.1	0.0	1.4
75	medium	7	0.2	0.1	25.4	0.2	0.1	0.9	0.1	0.0	1.1
75	medium	15	0.1	0.0	21.9	0.1	0.0	0.7	0.1	0.0	1.3
75	medium	25	0.1	0.0	20.5	0.1	0.0	0.5	0.1	0.0	1.6
75	large	7	4.7	4.4	29.0	4.7	4.4	1.1	0.3	0.0	0.7
75	large	15	0.7	0.4	25.8	0.7	0.4	1.0	0.3	0.0	1.1
75	large	25	0.1	0.0	23.7	0.1	0.0	0.7	0.1	0.0	1.6
100	small	10	0.1	0.0	63.6	0.1	0.0	1.8	0.1	0.0	3.1
100	small	20	0.1	0.0	56.1	0.1	0.0	1.4	0.1	0.0	3.4
100	small	33	0.1	0.0	53.5	0.1	0.0	1.1	0.1	0.0	3.4
100	medium	10	0.2	0.0	63.0	0.2	0.0	2.1	0.2	0.0	2.8
100	medium	20	0.1	0.0	60.1	0.1	0.0	1.8	0.1	0.0	2.9
100	medium	33	0.1	0.0	51.6	0.1	0.0	1.4	0.1	0.0	3.8
100	large	10	4.2	3.9	70.3	4.2	3.9	2.7	0.3	0.0	1.7
100	large	20	0.7	0.5	64.1	0.7	0.4	2.3	0.2	0.0	2.7
100	large	33	0.1	0.0	60.1	0.1	0.0	1.8	0.1	0.0	3.6

use of fractional variables. Aiming for a methodology to obtain strong dual bound, we exploited Dantzig-Wolfe decomposition to obtain an extended formulation, whose continuous relaxation is solved through column generation techniques. We proposed an ad-hoc pricing algorithm that decomposes the pricing problem into two subproblems, both solvable in pseudo-polynomial time. We integrated our bounding procedure into a branch-and-price framework including dual cuts, and implicit enumeration strategies to solve the problem to proven optimality.

Table 2 Results obtained within a time limit of 1 hour of computation.

Instance class			BC			CHAINBC			BPA		
$\|I\|$	weight	$\mathscr{F}$	s	$g(\%)$	$t(s)$	s	$g(\%)$	$t(s)$	s	$g(\%)$	$t(s)$
10	small	1	40	0.0	6.7	40	0.0	0.4	40	0.0	0.0
10	small	2	40	0.0	55.2	40	0.0	0.5	40	0.0	0.0
10	small	3	40	0.0	0.2	40	0.0	0.1	40	0.0	0.1
10	medium	1	40	0.0	105.3	40	0.0	5.8	40	0.0	0.0
10	medium	2	40	0.0	130.6	40	0.0	0.5	40	0.0	0.0
10	medium	3	40	0.0	0.6	40	0.0	0.2	40	0.0	0.1
10	large	1	38	3.2	509.3	40	0.0	49.5	40	0.0	0.0
10	large	2	36	1.0	465.7	40	0.0	36.5	40	0.0	0.0
10	large	3	38	0.7	9.0	40	0.0	1.1	40	0.0	0.1
25	small	2	30	3.1	115.5	30	0.3	11.7	40	0.0	0.3
25	small	5	36	1.9	28.4	34	0.0	5.8	40	0.0	0.7
25	small	8	40	0.0	4.8	39	0.0	4.3	40	0.0	2.1
25	medium	2	3	3.8	1269.8	10	1.5	672.3	40	0.0	0.2
25	medium	5	27	2.3	367.7	24	0.1	37.4	40	0.0	0.6
25	medium	8	39	0.6	129.8	31	0.1	24.0	40	0.0	1.9
25	large	2	0	7.1	-	0	5.8	-	40	0.0	0.1
25	large	5	2	3.4	2459.1	12	1.7	190.6	40	0.0	0.5
25	large	8	23	3.0	544.3	13	0.1	392.0	40	0.0	1.6
50	small	5	2	3.2	610.8	12	0.4	799.3	40	0.0	2.7
50	small	10	25	2.4	865.4	21	0.2	257.6	40	0.0	7.5
50	small	16	38	1.8	127.2	18	0.3	188.2	40	0.0	17.0
50	medium	5	0	3.9	-	1	0.9	2016.1	40	0.0	2.1
50	medium	10	0	2.8	-	3	0.5	1281.0	40	0.0	6.9
50	medium	16	23	2.1	896.0	12	0.2	382.1	40	0.0	18.4
50	large	5	0	7.4	-	0	4.3	-	40	0.0	1.0
50	large	10	0	5.3	-	0	1.5	-	40	0.0	7.6
50	large	16	1	3.2	1484.1	3	0.6	1434.6	40	0.0	17.8
75	small	7	0	4.9	-	2	0.9	325.9	40	0.0	9.9
75	small	15	3	3.0	3276.4	10	0.3	724.2	40	0.0	27.0
75	small	25	35	1.7	357.1	16	0.1	465.1	40	0.0	79.4
75	medium	7	0	8.8	-	0	1.6	-	40	0.0	7.6
75	medium	15	0	5.1	-	1	1.0	1074.3	40	0.0	23.8
75	medium	25	7	2.2	1243.5	10	0.4	809.8	40	0.0	71.7
75	large	7	0	12.5	-	0	19.8	-	40	0.0	4.5
75	large	15	0	8.0	-	0	14.1	-	40	0.0	30.1
75	large	25	0	4.6	-	0	0.9	-	40	0.0	77.1
100	small	10	0	7.5	-	1	1.2	953.3	40	0.0	25.7
100	small	20	0	3.9	-	10	0.5	956.3	40	0.0	73.4
100	small	33	25	1.4	638.6	17	0.2	441.3	40	0.0	178.8
100	medium	10	0	9.5	-	0	6.7	-	40	0.0	30.1
100	medium	20	0	6.8	-	0	1.3	-	40	0.0	69.4
100	medium	33	0	2.9	-	5	0.6	1094.1	40	0.0	210.5
100	large	10	0	17.6	-	0	37.9	-	38	0.1	25.4
100	large	20	0	11.1	-	0	24.0	-	40	0.0	70.2
100	large	33	0	7.0	-	0	1.2	-	40	0.0	230.2

After an extensive experimental campaign, our algorithm was proven to be able to solve 99% of our dataset within few minutes of computation, while state-of-the-art solvers failed to solve even very small instances.

References

1. Achterberg, T.: Scip: solving constraint integer programs. Mathematical Programming Computation **1**(1), 1 – 41 (2009)

2. Casazza, M., Ceselli, A.: Mathematical programming algorithms for bin packing problems with item fragmentation. Computers & Operations Research **46**, 1 – 11 (2014)
3. Casazza, M., Ceselli, A.: Column generation for the variable cost and size bin packing problem with fragmentation. Electronic Notes in Discrete Mathematics **55**, 61 – 64 (2016)
4. Casazza, M., Ceselli, A.: Exactly solving packing problems with fragmentation. Computers & Operations Research **75**, 202 – 213 (2016)
5. Correia, I., Gouveia, L., da Gama, F.S.: Solving the variable size bin packing problem with discretized formulations. Computers & Operations Research **35**(6), 2103 – 2113 (2008). Part Special Issue: OR Applications in the Military and in Counter-Terrorism
6. CPLEX development team: Ibm ilog cplex optimization studio: Cplex user's manual - version 12 release 6. Tech. rep., IBM corp. (2014)
7. Crainic, T.G., Perboli, G., Rei, W., Tadei, R.: Efficient lower bounds and heuristics for the variable cost and size bin packing problem. Computers & Operations Research **38**(11), 1474 – 1482 (2011)
8. Dantzig, G., Wolfe, P.: Decomposition principle for linear programs. Operations research **8**, 101–111 (1960)
9. LeCun, B., Mautor, T., Quessette, F., Weisser, M.A.: Bin packing with fragmentable items: Presentation and approximations. Theoretical Computer Science **602**, 50 – 59 (2015)
10. Mandal, C., Chakrabarti, P., Ghose, S.: Complexity of fragmentable object bin packing and an application. Computers & Mathematics with Applications **35**(11), 91 – 97 (1998)
11. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, Inc., New York, NY, USA (1990)
12. Menakerman, N., Rom, R.: Bin Packing with Item Fragmentation, pp. 313–324. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
13. Naaman, N., Rom, R.: Packet scheduling with fragmentation. In: Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, pp. 427–436 vol.1 (2002)
14. Secci, S., Ceselli, A., Malucelli, F., Pattavina, A., Sansò, B.: Direct optimal design of a quasi-regular composite-star core network. In: IEEE Proc. of DRCN, pp. 7–10 (2007)
15. Shachnai, H., Tamir, T., Yehezkely, O.: Approximation schemes for packing with item fragmentation. Theory of Computing Systems **43**(1), 81–98 (2008)
16. Shachnai, H., Yehezkely, O.: Fast Asymptotic FPTAS for Packing Fragmentable Items with Costs, pp. 482–493. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
17. Vanderbeck, F.: Implementing Mixed Integer Column Generation, pp. 331–358. Springer US, Boston, MA (2005)