Solving robust bin-packing problems with a branch-and-price approach

Xavier Schepler, André Rossi, Evgeny Gurevsky, Alexandre Dolgui

Version of Record: https://www.sciencedirect.com/science/article/pii/S0377221721004690
Manuscript_8a907edd7b5f5be37fe0ca6d060309ee

Solving robust bin-packing problems with a branch-and-price approach

Xavier Schepler[a,b,*], André Rossi[a,c], Evgeny Gurevsky[d], Alexandre Dolgui[e]

[a] *LERIA, Université d'Angers, France*
[b] *Recommerce R&D, France*
[c] *LAMSADE, Université Paris-Dauphine, PSL, France*
[d] *LS2N, Université de Nantes, France*
[e] *IMT Atlantique, LS2N, Nantes, France*

Abstract

One-dimensional bin-packing is a well-known combinatorial optimization problem which is strongly NP-hard. It consists of allocating a given set of items of different sizes into bins of the same capacity to minimize the number of bins used. The capacity of each bin cannot be exceeded. This paper deals with some variants of this problem to take into account the cases when there are items with uncertain sizes. The goal is to obtain robust solutions taking into account possible variations of item sizes around their nominal values. First, two robust approaches are considered which are based on a stability radius calculation, to ensure that the *stability radius*, measured either with the Manhattan or Chebyshev norm, is not below a given threshold. Then, a complementary robust approach is applied which is based on a *relative resiliency* calculation. To solve to optimality these robust variants of the bin-packing problem, a compact 0-1 linear programming formulation, which is also valid for the standard bin-packing problem, is introduced. Then, a Dantzig-Wolfe decomposition is suggested in order to provide a set-cover reformulation with a stronger linear relaxation, but an exponential number of columns. Finally, to obtain integer optimal solutions, a branch-and-price algorithm is developed, whose linear relaxation of the set-cover formulation is solved by a dynamic column generation. Numerical experiments are conducted on adapted benchmark sets from the literature. The performance of the branch-and-price algorithm allows us to investigate what protection against uncertainty is offered by each approach, and at which cost of robustness.

Keywords: Packing, Robustness, Sensitivity analysis, Uncertainty, Stability radius, Relative resiliency, Dantzig-Wolfe decomposition, Column generation, Branch-and-price

*Corresponding author
Email address: xavier.schepler@gmail.com (Xavier Schepler)

1. Introduction

For the standard bin-packing (BP) problem, one-dimensional items of different sizes have to be assigned to a minimum number of identical one-dimensional bins. The size of each item and the capacity of a bin are assumed to be known in advance. Nevertheless, in real-life applications of BP, the sizes of items often have random deviations from the nominal values. For example, in hospital administration (see Cardoen et al., 2010; Lamiri et al., 2008), one of the most important optimization issues which can be modeled as BP problems, is the daily planning of surgeries. The surgery durations are likely to change. A smaller number of operating rooms used per day reduces the cost of dealing with their preparation. The context of BP under uncertainty appears also in the domain of supply-chain management. For example, for the problem of logistics capacity planning (see Crainic et al., 2016), where it is necessary to predict four to six months in advance the allocation of goods to a minimum number of containers of different capacities. The volumes of delivered goods have random deviations around the planned values. BP models are also used in the design of paced assembly lines. The problem is known in the literature as an assembly balancing problem of type 1 (see Boysen et al., 2007; Battaïa and Dolgui, 2013), where a given set of production tasks, some involving issues of precedence, has to be assigned to a minimum number of identical workstations. The so-called cycle time constraint expresses a limit on a station capacity. In the case of non-fully automated lines, there is a subset of manual tasks, whose processing times may vary depending on operator skills and random performances. The assembly line balancing problem of type 1 is a BP problem with precedence constraints. Another case of BP problems under uncertainty is the one dimensional cutting stock problem, as in Alem et al. (2010), where the demand or item sizes can be uncertain. A robust machine availability problem in a parallel machine environment with uncertain task durations in Song et al. (2018) is another example. The latter two articles are discussed in more detail in Section 3, dedicated to the literature review.

Probabilistic models are often used to consider uncertainty. Nevertheless, this approach is not always possible because of the absence of reliable historical data in order to establish adequate probability distributions as, for example, in the case of the initial design of the aforementioned assembly lines. Only a subset of tasks with uncertain processing times can be identified by decision makers at the line design stage. In order to overcome this difficulty, Rossi et al. (2016), motivated by the work of Sotskov et al. (2006), proposed a new approach, based on the stability radius optimization. For any given solution, its stability radius is obtained as the maximal magnitude of variations from their nominal values of the uncertain task times for which the solution feasibility (or optimality) is preserved. In other words, studying the simple assembly line balancing problem with a fixed number of

workstations, the authors search for solutions which maximize the stability radius. The well-known norms ℓ_1 and ℓ_∞ are used to calculate the stability radius. First, the authors show that if the ℓ_1 norm is used, then the minimum *idle time*[1] among all the workstations having at least one uncertain task, has to be maximized. Second, they also show that if the ℓ_∞ norm is employed, a duration equal to the stability radius, added to the duration of each uncertain task, has to be maximized. Thus, from a managerial point of view, it is necessary to search for a feasible task assignment having the maximal value of the mentioned above indicators, since this leads to the greatest protection against uncertainty.

In this paper, we develop a branch-and-price method applied to the problems, which are inverse to those addressed in Rossi et al. (2016). By considering a BP problem, therefore excluding precedence constraints, the problem consists of allocating a given set of items to a minimum number of bins so that the stability radius of the solution is not less than a specified threshold, which determines a desired level of robustness. Then, to overcome a possible limitation of the previous approach, in which the variation of an uncertain size is not related to its nominal value, another robust approach based on the concept of relative resiliency proposed by Pirogov (2019), is developed. The relative resiliency of a solution is the proportion by which each uncertain size can be increased without losing solution feasibility.

Hereafter, the robust BP problem ensuring that the stability radius calculated for the ℓ_1 norm (resp. the ℓ_∞ norm) is not less than r is denoted as $RBP_1(r)$ (resp. $RBP_\infty(r)$), and the robust BP problem ensuring the relative resiliency is not less than α is denoted as $RBP_{rr}(\alpha)$. These three problems can be simply stated as follows.

We want to pack a given set of one-dimensional items whose nominal sizes are known into a minimum number of one-dimensional identical bins of the same given capacity. A known subset of these items have uncertain sizes, which can vary after the allocation of items to bins. The solution must remain feasible for real item sizes, provided that the variations of uncertain sizes are bounded as follows:

- with $RBP_\infty(r)$, each uncertain size can be increased by a maximum of r,
- with $RBP_{rr}(\alpha)$, each uncertain size w_i can be increased by a maximum of αw_i,
- with $RBP_1(r)$, in any bin with at least one uncertain-size item, the sum of all uncertain sizes' variations is at most r.

[1]Idle time is the difference between the cycle time and the effective load of the considered workstation.

Given these conditions, a solution to any instance of $RBP_\infty(r)$ (resp. $RBP_{rr}(\alpha)$) can be obtained by solving an instance of BP, in which each uncertain size w_i is increased by r (resp. αw_i), to reserve free space. Therefore, $RBP_\infty(r)$ and $RBP_{rr}(\alpha)$ can both be straightforwardly reduced to BP. However, this is not the case with $RBP_1(r)$, as it requires a quantity of free space not lesser than r in each bin containing at least one item of uncertain size.

Note that, a solution to BP has a stability radius (resp. relative resiliency) at least equal to r (resp. α) if and only if a quantity r (resp. αw_i) of free space is reserved for each item i of uncertain size w_i in its assigned bin. Similarly, a solution to BP has a stability radius at least equal to r if and only if a quantity r of free space is reserved in each bin containing at least one uncertain item. Formal definitions of stability radius, relative resiliency, and proofs of the properties stated above are provided in Appendix A.

The remainder of this paper is organized as follows. Section 2 provides a literature review about column-generation based approaches for variants of BP under uncertainty and branch-and-price algorithms for BP. Section 3 presents the details of the adapted branch-and-price method for the proposed set-cover formulation valid for $RBP_1(r)$, $RBP_\infty(r)$, $RBP_{rr}(\alpha)$ and BP. Section 4 is devoted to numerical experiments, in which we evaluate the performance of the proposed branch-and-price algorithm and we investigate what protection against uncertainty is offered by each approach and at which cost of robustness. Section 5 concludes the paper and gives some perspectives for future work.

2. Literature review

We first consider two papers introducing column-generation based algorithms to tackle variants of BP under uncertainty, then the literature about branch-and-price algorithms designed to solve the set-cover formulation of BP provided in Section 3.2.

Song et al. (2018) address the robust machine availability problem in a parallel machine environment, which in its deterministic version coincides with BP. Using the methodology proposed by Bertsimas and Sim (2004), they introduce a robust version with budgeted uncertainty, where at most Γ jobs can deviate from their nominal processing times, which are assumed to belong to symmetric and bounded intervals. They propose a set-cover formulation with an exponential number of columns. The formulation is solved by a branch-and-price algorithm, with the Ryan and Foster branching scheme, and zero-suppressed binary decision diagrams to handle a pricing problem more complicated than the one occuring with BP. They perform numerical experiments on a new set of randomly generated instances. Results show that their algorithm can handle instances with up to

180 items within a time limit of 1200 seconds, provided that either the value of Γ is small, or the size of each item is in the range $\{1, \ldots, 20\}$.

Alem et al. (2010) address a stochastic variant of the one dimensional cutting stock problem, where demand is a random variable. The authors propose a two stage stochastic non-linear program, where the first stage decisions are the numbers of objects to be cut according to a cutting pattern, and the second stage decisions are the numbers of holding or backordering items. The objective is to minimize the expected costs due to waste and holding or backordering penalties. A column-generation based algorithm is introduced to solve the linear relaxation of the formulation, and heuristics allow integer solutions to be obtained. Numerical experiments are performed to investigate the value of the stochastic solution and the expected value with perfect information. The results show that the two-stage stochastic approach represents a better modeling framework than the alternative wait-and-see and expected value frameworks, even under small variations in the parameters of the problem.

There is abundant literature about branch-and-price algorithms, including general studies, for example by Barnhart et al. (1998), Vanderbeck (2000), and Desrosiers and Lübbecke (2011), as well as many other application-related studies, in various domains, such as cutting and packing (Delorme et al. (2016)), vehicle routing (Fukasawa et al. (2006)), production planning (Alfandari et al. (2015)), etc. The theories and concepts behind branch-and-price algorithms are covered in the Ph.D. thesis by Gamrath (2010).

We focus on two branch-and-price algorithms similar to the one proposed in Section 3 designed to solve the set-cover formulation of BP, which corresponds to the set-cover formulation proposed in this paper, when there is either no uncertain item or all items are uncertain, or r is equal to zero. These are the ones by Vance et al. (1994) and by Gamrath et al. (2016).

The first branch-and-price algorithm for the set-cover formulation of BP was proposed by Vance et al. (1994). The first-fit decreasing heuristic initializes the solving process with a primal solution. The branch-and-price algorithm relies on the Ryan and Foster branching scheme, which selects one of the fractionally packed pairs of items for branching, and performs branching so that either the items are packed in the same bin, or in two different bins. In the first case, when items have to be packed in the same bin, the structure of the subproblem is preserved, as it suffices to merge the items. In the second case, the structure of the subproblem is altered, with the presence of a *conflict* constraint, forbidding the items in conflict to be packed together. When considering pairs of items for branching, the one with the largest total size is selected. Nodes of the search tree without any conflict constraint are explored first, as their subproblem can be solved by a more efficient algorithm.

Vance et al. (1994) proposed solving the subproblem without conflict constraint, that is, the pseudo-polynomial 0-1 knapsack problem, with the Horowitz and Sahni (1974) algorithm, having a worst case temporal complexity in $\mathcal{O}(\min\{2^{n/2}, n \cdot W\})$, potentially better than the one of the classical dynamic programming algorithms, in $\mathcal{O}(n \cdot W)$, described by Kellerer et al. (2004). When each item is in conflict with at most one other item, Vance et al. (1994) use an adapted version of the algorithm by Horowitz and Sahni (1974). Otherwise, in the presence of conflict constraints, the subproblem becomes strongly NP-hard, and it is addressed with an integer linear program solver.

The branch-and-price algorithm proposed by Gamrath et al. (2016) also uses a branching scheme based on the Ryan and Foster branching scheme. Nevertheless, it differs from the one by Vance et al. (1994) on several points. First, one of the most unfeasible pairs of items is selected for branching, that is, the one which provides a sum value closest to 0.5. Second, the node which is selected for exploration is the one which provides the best estimation value of the progress toward integer feasibility, relative to its degradation of the objective function. Third, since this branch-and-price algorithm is provided as an example with the SCIP Optimization Suite, it relies on its integer linear program solver to address the subproblem. Last, for the same reason, it uses primal heuristics of the SCIP Optimization Suite, described in Berthold (2008) and Achterberg et al. (2012). This includes more than forty very fast heuristics, generally requiring only a few milliseconds, even with the largest BP instances. They are called repetitively during the exploration of the search tree. The branch-and-price algorithm that we propose is presented in the next section, and some of the concepts involved, such as dynamic column generation, are covered in more detail.

3. A branch-and-price algorithm

We are given a set K of one-dimensional identical bins of capacity W, a set $V = \{1, 2, \ldots, n\}$ of one-dimensional items, a nominal size w_i for each item $i \in V$, and a set $\widetilde{V} \subseteq V$ of items of uncertain size. An allocation of the set of items to the set of bins is called a feasible solution to BP if each item is assigned to exactly one bin such that the capacity constraints are satisfied. We use the following decision variables: x_{ik} is set to one if item i is assigned to bin k, and zero otherwise, and y_k is set to one if bin k is used and zero otherwise.

We introduce a branch-and-price algorithm for $RBP_1(r)$ derived from the ones proposed by Vance et al. (1994) and by Gamrath et al. (2016) for BP. This algorithm can also be used for BP, which corresponds to $RBP_1(r)$ when there is no uncertain items, and, therefore, for $RBP_\infty(r)$ and $RBP_{rr}(\alpha)$, as explained below.

$RBP_\infty(r)$ can be formulated as follows.

$$\min \quad \sum_{k \in K} y_k \tag{1a}$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} \geq 1, \quad \forall i \in V, \tag{1b}$$

$$\sum_{i \in \widetilde{V}} (w_i + r) x_{ik} + \sum_{i \in V \setminus \widetilde{V}} w_i x_{ik} \leq W y_k, \quad \forall k \in K, \tag{1c}$$

$$y_k \in \{0, 1\}, \quad \forall k \in K, \tag{1d}$$

$$x_{ik} \in \{0, 1\}, \quad \forall i \in V, \quad \forall k \in K. \tag{1e}$$

Where (1a) minimizes the number of open bins, (1b) assigns each item to at least one bin, (1c) enforces the bin capacity. Here, the size of all the uncertain items has been increased by the positive constant r, as per Property 4 presented in Appendix A, and (1d)–(1e) are domain constraints.

$RBP_{rr}(\alpha)$ can be formulated using the same BP model, but where constraints (1c) should be replaced with:

$$\sum_{i \in \widetilde{V}} w_i (1 + \alpha) x_{ik} + \sum_{i \in V \setminus \widetilde{V}} w_i x_{ik} \leq W y_k, \quad \forall k \in K.$$

In this constraint, the size of all the uncertain items is multiplied by $1 + \alpha$, where α is a positive constant, as per Property 5 presented in Appendix A.

3.1. Compact formulation of $RBP_1(r)$

The compact formulation of $RBP_1(r)$ uses an additional decision variable: a_k is set to one if bin k contains at least one uncertain-size item, and zero otherwise.

$RBP_1(r)$ can be formulated as follows.

$$\min \quad \sum_{k \in K} y_k \tag{2a}$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} \geq 1, \quad \forall i \in V, \tag{2b}$$

$$x_{ik} \leq a_k, \quad \forall i \in \widetilde{V}, \quad \forall k \in K, \tag{2c}$$

$$\sum_{i \in V} w_i x_{ik} + r a_k \leq W y_k, \quad \forall k \in K, \tag{2d}$$

$$y_k \in \{0, 1\}, \quad \forall k \in K, \tag{2e}$$

$$a_k \in \{0, 1\}, \quad \forall k \in K, \tag{2f}$$

$$x_{ik} \in \{0, 1\}, \quad \forall i \in V, \quad \forall k \in K. \tag{2g}$$

Constraints (2b) are the same as (1b). Constraints (2c) give to each variable a_k the value 1 if bin k contains at least one uncertain-size item. Constraints (2d) represent the bin capacity, taking into account the minimum desired level of robustness r for the bins having uncertain-size items. This constraint is also a natural consequence of Property 6, presented in Appendix A. More specifically, this property pertains to bins that accommodate uncertain items. In the present formulation of $RBP_1(r)$, these bins are open ($y_k = 1$) and such that $a_k = 1$. Objective (2a) is to minimize the number of bins used.

Note that, BP is a particular case of $RBP_1(r)$, where there is no uncertain item ($\widetilde{V} = \emptyset$), or all items are uncertain ($V = \widetilde{V}$), or, alternatively $r = 0$. When there is no uncertain item, variables a_k can be set to 0, and we obtain a formulation of BP. When all items are uncertain, or $r = 0$, variables a_k can be set to 1, and we also obtain a formulation of BP.

Besides, the linear programming relaxation of this compact formulation is expected to be weak, as shown by Vance et al. (1994) for their formulation of BP, which we extended. To check if a state-of-the-art solver is able to solve this compact formulation, we performed numerical experiments on adapted instances of Falkenauer (1996). We added 20% of uncertain-size items and we set r equal to 10% of the bin capacity. Only 30 of the 80 adapted instances were solved by IBM CPLEX 12.7 within a time limit of one hour per instance, which motivated us to develop a more efficient approach.

3.2. Set-cover reformulation of $RBP_1(r)$

Another possibility is to use a set-cover formulation, which is obtained by applying the Dantzig-Wolfe decomposition principle (Dantzig and Wolfe (1960)), in order to obtain in this case a much tighter lower bound on the optimal value of bins.

Once the constraints (2b) are dualized in a Lagrangian way (Lemaréchal (2007)), subsystem (2c)-(2d) decomposes into a subproblem for each bin $k \in K$. Let $\mathcal{B}$ be the family of the subsets of items that can be fit into one bin with respect to uncertain sizes and to the threshold r on stability radius ρ_1. Each subset $B \in \mathcal{B}$ is defined by an indicator vector x^B ($x_i^B = 1$ iff item $i \in V$ is in set B) and associated with a binary variable λ_B taking value 1 if the corresponding subset of items is selected to fill one bin.

The reformulation is:

$$\min \quad \sum_{B \in \mathcal{B}} \lambda_B \tag{3a}$$

$$\text{s.t.} \quad \sum_{B \in \mathcal{B}} x_i^B \lambda_B \geq 1, \quad \forall i \in V, \tag{3b}$$

$$\sum_{B \in \mathcal{B}} \lambda_B \leq K \tag{3c}$$

$$\lambda_B \in \{0,1\}, \quad \forall B \in \mathcal{B}. \tag{3d}$$

Here, constraints (3b) replace constraints (2b), constraints (3c) limit the number of used bins to at most K, and all other constraints of the compact formulation are built into the definition of feasible sets $B \in \mathcal{B}$ of items. This problem will be referred to as the master problem. When $\widetilde{V} = \emptyset$ or $\widetilde{V} = V$ or $r = 0$, we obtain the set-cover formulation of BP, for which it is conjectured that the difference between the linear relaxation value at root node and the optimal solution value is always less than or equal to two (see, for example, Scheithauer and Terno, 1997).

The set-cover formulation, which has an exponential number of columns, is addressed by a branch-and-price approach. Its principles are described in Section 3.3. At each node of a branch-and-bound tree, its linear relaxation is solved by dynamic column generation. The master problem is restricted to a subset of columns, and its linear relaxation is solved by the simplex method. Then, a *pricing subproblem* is solved to check for the existence of an improving column, with a negative reduced cost. It consists in solving the following robust variant of the knapsack problem, denoted as $PS^{(LP)}$:

$$\max \quad \sum_{i \in V} \pi_i z_i \tag{4a}$$

$$\text{s.t.} \quad z_i \leq a, \quad \forall i \in \widetilde{V}, \tag{4b}$$

$$\sum_{i \in V} w_i z_i + ra \leq W, \tag{4c}$$

$$z_i \in \{0,1\}, \quad \forall i \in V, \tag{4d}$$

$$a \in \{0,1\}. \tag{4e}$$

The formulation of the pricing subproblem is related to a vector of dual variables π corresponding to constraints (3b) of the set-cover reformulation. A binary variable z_i indicates whether item i is selected. A binary variable a indicates whether the knapsack contains at least one uncertain-size item or not. Clearly, the feasible solutions to this problem are the feasible subsets $B \in \mathcal{B}$ of items.

3.3. Dynamic column generation at root node

The number of variables of the set-cover formulation grows exponentially with the number of items. Hence, it is only possible to generate all of its columns when the number of items is small. Otherwise, with a large number of items, the well-known principle of dynamic column generation can be applied. Dynamic column generation allows the linear relaxation of the set-cover formulation to be solved, as proposed by Dantzig and Wolfe (1960) for linear programs, considering, implicitly, the whole set of columns. An iteration of dynamic column generation starts with solving the linear relaxation of the set-cover formulation restricted to a relatively small subset of columns. The first iteration begins with a column pool constituted of columns corresponding to subsets of $\mathcal{B}$ containing exactly one item, and of columns in the solution provided by the adapted first-fit decreasing heuristic, described below in Section 3.6.1. Solving the restricted master problem with the Simplex method provides values of dual variables π_i, for all $i \in V$.

Then, to check whether an improving column exists, reduced cost pricing is performed, by solving $PS^{(LP)}$, for the current values of dual variables. If its optimal objective value is strictly larger than one, the corresponding improving column, with a negative reduced cost, is added to the current restricted master problem, providing a new restricted master problem, and a new iteration of dynamic column generation starts. If not, no improving column exists, and the current feasible solution to the linear relaxation of the set-cover formulation is optimal.

$PS^{(LP)}$ is a robust variant of the 0-1 knapsack problem. It can be solved in pseudo-polynomial time, by solving two instances of the 0-1 knapsack problem, as shown below.

Property 1. *$PS^{(LP)}$ can be solved in $\mathcal{O}(n \cdot W)$, by solving two instances of the 0-1 knapsack problem.*

Proof. First, the value of binary variable a is set to 0 (the bin contains no uncertain-sized item), and the resulting 0-1 knapsack problem is solved in $\mathcal{O}(n \cdot W)$, with the classical dynamic programming algorithm (see Toth (1980)). Second, the same is done for $a = 1$ (the bin is allowed to contain at least one uncertain-sized item). Then, one of the two solutions which provides the best value is selected, and it is an optimal solution to $PS^{(LP)}$. Hence, solving $PS^{(LP)}$ requires $\mathcal{O}(2 \cdot n \cdot W) = \mathcal{O}(n \cdot W)$ operations. □

Note that, by ordering the items such that certain-sized items are followed by uncertain-sized items, and by adapting the classical dynamic programming algorithm (see Toth (1980)) so that, when uncertain-sized items are reached, the knapsack capacity is decreased by r, it is possible to

solve $PS^{(LP)}$ with a single call to this adapted algorithm. However, we use two calls to the COMBO algorithm (Martello et al. (1999)) to deal with $PS^{(LP)}$ and solving two knapsack problems.

As its name suggests, the COMBO algorithm is a combination of several techniques. One of the central ideas upon which it relies is that a small subset of items is generally enough to form an optimal solution. Therefore, the COMBO algorithm restricts the search to a subset of items denoted as the core, which is expanded when needed. Besides this, the algorithm makes use of upper and lower bounds, which allow to fathom dominated states in the list-based dynamic programming procedure, and to terminate the search earlier.

When the COMBO algorithm is called to solve the second instance of the 0-1 knapsack problem, it takes advantage of the lower bound provided by the solution to the first one. It generally enables us to solve the robust 0-1 knapsack subproblem faster than the adapted dynamic programming algorithm. Note that, for a pricing subproblem occurring when $\widetilde{V} = \emptyset$, COMBO is called only one time (for $a = 0$).

Finally, we use the bound proposed by Farley (1990) to terminate column generation early when

$$\lceil v_{\text{RMP}} \rceil = \left\lceil \frac{v_{\text{RMP}}}{v_{\text{PS}}} \right\rceil,$$

where v_{RMP} is the linear relaxation value of the current restricted master problem and v_{PS} is the optimal solution value to the current pricing subproblem. This bound is also used by Vance et al. (1994) and Song et al. (2018). It was shown to be effective in reducing tailing off effects.

3.4. Branching scheme

Dynamic column generation provides solutions to the linear relaxation of the set-cover formulation. However, its solution generally does not satisfy integrality constraints. When some of the variables of the solution to the linear relaxation of Problem (3) have non-integral values, branching is performed. We use the branching rule described by Ryan and Foster (1981), which relies on the following property.

Property 2. *Let $\mathcal{B}_{i,j} \subset \mathcal{B}$ be the set of patterns that contains both items i and j. A solution to the linear relaxation of the set-cover formulation contains variables with non-integral values if and only if there exists an item pair $\{i, j\}$ such that: $0 < \sum_{\mathcal{B}_{i,j}} \lambda_B < 1$ (Ryan and Foster (1981)).*

When the solution to the linear relaxation of the set-cover formulation contains variables with non-integral values, we select one of the item pairs $\{i, j\}$ with the most unfeasible value, that is, which minimizes $|0.5 - \sum_{\mathcal{B}_{i,j}} \lambda_B|$. At least one such pair is guaranteed to exist, according to Property 2.

Then, two child nodes are created. In the left node, items i and j have to be in the same bin: the branching constraint is $z_i = z_j$. In the right node, items i and j have to be in different bins: the branching constraint is $z_i + z_j \leq 1$. In each of the two nodes, the variables of the restricted master problem corresponding to columns which do not satisfy its branching constraints are removed. Each branching constraint is directly added to the subproblem of its node, so that such columns are not generated again. The problem of each node is solved by column generation, and branching occurs as with the root node. The search tree is explored by a depth-first search procedure with restart, in which the best-bound node is selected periodically for a new start of depth-first search.

In Vance et al. (1994), a different item pair selection strategy is proposed: one of the $\{i, j\}$ with the largest total size, but lower than the bin capacity, is selected for branching. We performed numerical experiments with the 50 difficult AI instances introduced by Delorme et al. (2016) to compare this node selection strategy with the one we propose. These instances do not allow free space in any optimal solution. The time limit was set to 3600 seconds. Results are reported in Table 1. Column "*Strategy*" provides the node-selection strategy. Column "*Avg.* `CPU`*, (s.)*" gives the average solution time in seconds. Column "`#OPT`" provides the number of proven optimal solutions obtained. Column "`#NODES`" gives the average number of opened nodes in the search tree.

Strategy	`#OPT`	Avg. `CPU`, (s.)	Avg. `#NODES`
Most unfeasible	32	1337	8420
Largest sum	12	2796	11444

Table 1: Comparison of two node-selection strategies

The "*Most unfeasible*" strategy of the proposed branch-and-price algorithm shows faster convergence than the "*Largest sum*" strategy with these instances. These instances require the exploration of a relatively large search tree, and the choice of this strategy is therefore important with them. In the case of a non-optimal solution, only one extra bin was always used.

3.5. Subproblems after branching

Imposing branching constraints $z_i = z_j$ on pairs $\{i, j\}$ of items in a 0-1 robust knapsack subproblem preserves its structure. Each of these constraints ensure that either both or none of the items of its pair are selected. Hence, items i and j can be merged into an item of weight $w_i + w_j$ and value $\pi_i + \pi_j$. The resulting item has an uncertain size if i or j has an uncertain size, otherwise it is a certain-size item. This subproblem is denoted as $PS^{(left)}$ and it is equivalent to $PS^{(LP)}$.

However, imposing branching constraints $z_i + z_j \leq 1$ on pairs $\{i, j\}$ of items (*conflict* constraints,

also called disjunctive constraints) to a 0-1 robust knapsack subproblem alters its structure. It turns the subproblem into a robust 0-1 knapsack problem with conflict constraints on items, denoted as $PS^{(right)}$. The non-robust variant of this problem is referred in the literature to as 0-1 knapsack problem with conflicts. The 0-1 knapsack problem with conflicts is strongly NP-hard, and the more general robust variant is therefore also strongly NP-hard. Property 1 can be straightforwardly adapted to $PS^{(right)}$, which is addressed by solving two instances of the 0-1 knapsack problem with conflicts. The optimal solution value to the first $PS^{(right)}$ instance (for $a = 0$), is used as a lower bound for the second instance (for $a = 1$). Note that, for a pricing subproblem occurring when $\widetilde{V} = \emptyset$, this algorithm is called only one time (for $a = 0$).

We solve the 0-1 knapsack problem with conflicts with the dedicated branch-and-bound algorithm by Sadykov and Vanderbeck (2013), which combines the classic depth-first search based branch-and-bound algorithm for the 0-1 knapsack problem by Kellerer et al. (2004) with the enumeration algorithm for solving the maximum clique problem (or the independent set problem) by Östergård (2002). Upper bounds are computed at each node of the search tree by solving the linear relaxation of the residual knapsack problem, ignoring conflict constraints, with the well-known greedy algorithm for the 0-1 knapsack problem. This algorithm sorts the items in decreasing order of utility (*i.e.* the ratio value over weight), and then proceeds to insert them into the knapsack. A limit is set on the number of nodes, which is active only if the current primal bound is greater than one, that is, if an improving column has been found. The branch-and-bound terminates if this limit is reached and if one improving column is available. Otherwise, the solving process goes on, to obtain an improving column, or to prove that no such column exists.

Finally, we introduce the following dominance rule between items for $PS^{(right)}$, which is an extension of classical dominance rules.

Property 3. *Let $C_i \subset V$ be the set of items in conflict with $i \in V$, i.e., for each $j \in C_i$ we have $x_i + x_j \leq 1$. If the following four conditions hold for any $j \in C_i$, then item j is dominated by item i, and can be removed from the instance of the robust 0-1 knapsack problem with conflicts:*

1. $i \in \widetilde{V} \wedge j \in \widetilde{V}$ *or* $i \notin \widetilde{V} \wedge j \notin \widetilde{V}$ *or* $i \notin \widetilde{V} \wedge j \in \widetilde{V}$,
2. $\pi_i \geq \pi_j$,
3. $w_i \leq w_j$,
4. $C_i \setminus \{j\} \subseteq C_j \setminus \{i\}$.

Proof. We show that, if the conditions are verified, any feasible solution s containing item j can be substituted by another solution s' in which item j is replaced by item i, such that the value of s' is greater than or equal to the value of s.

First, solution s' is feasible, regarding (1) the slack space r required in the presence of at least one uncertain item, as $i \in \widetilde{V} \wedge j \in \widetilde{V}$ or $i \notin \widetilde{V} \wedge j \notin \widetilde{V}$ or $i \notin \widetilde{V} \wedge j \in \widetilde{V}$, (2) the total item size, as $w_i \leq w_j$, and (3) the conflicts between items, as $C_i \setminus \{j\} \subseteq C_j \setminus \{i\}$. Second, solution s' has a value greater than or equal to solution s since $\pi_i \geq \pi_j$.

Hence, if the conditions are verified, it is not necessary to consider any solution containing j, which can be removed from the instance of the robust 0-1 knapsack problem with conflicts. □

This dominance rule allows to eliminate some items of an instance of $PS^{(right)}$ in a preprocessing phase. It is not mentioned in the preceding studies by Vance et al. (1994) and Sadykov and Vanderbeck (2013), which uses the same branching rule, that conducts to a non-robust 0-1 knapsack problem with conflicts as subproblem.

We performed numerical experiments with the 50 difficult AI instances introduced by Delorme et al. (2016) to evaluate the impact of the proposed dominance rule. They do not allow free space in any optimal solution. The time limit was set to 3600 seconds. Results are reported in Table 2. Column "*Dominance rule*" indicates whether the dominance rule is used or not. Column "*Avg.* `CPU`*, (s.)*" gives the average solution time in seconds. Column "#`OPT`" provides the number of proven optimal solutions obtained. Column "#`NODES`" gives the average number of opened nodes in the search tree.

Dominance rule	#`OPT`	Avg. `CPU`, (s.)	Avg. #`NODES`
Yes	32	1337	8420
No	24	1879	6427

Table 2: Impact of the dominance rule

The proposed dominance rule enables a faster convergence for the algorithm, while exploring a larger number of nodes, as subproblems are solved faster after the reduction of the number of items they contain, and this reduction is done efficiently. In the case of a non-optimal solution, only one extra bin was always used.

3.6. Primal heuristics

We use the following primal heuristics, which have an important role in the efficiency of the proposed branch-and-price-algorithm, because they generally provide optimal or near optimal solutions earlier than the branching procedure would.

3.6.1. Adapted first-fit decreasing heuristic

The FFD (First-Fit Decreasing heuristic) for BP can be stated as follows: items are sorted by non-increasing weights, and placed into the first bin in that order where it is possible. If no bin with enough free space is available, a new bin is used. FFD uses no more that $11/9\ OPT + 1$ bins for BP, where OPT denotes the optimal number of required bins, as shown in Yue (1990).

FFD is simply adapted to $RBP_1(r)$ as follows: first, FFD is applied to uncertain-sized items only, with the capacity of the used bins decreased by r, producing a partial solution; second, FFD is applied to certain-sized items, using the residual bin capacities of the partial solution, and using new bins when needed.

The main idea here is to pack together uncertain-sized items, as the presence of only one uncertain-sized item in a bin decreases its capacity by r. Hence, in an optimal solution, uncertain-sized items tend to be packed together as much as possible. The adapted FFD for $RBP_1(r)$ is used to initiate the branch-and-price algorithm with a primal solution, providing both a primal bound and columns for the first restricted master problem.

3.6.2. Integer programming based heuristics

We use all the integer programming based heuristics provided with the SCIP Optimization Suite, upon which our branch-and-price algorithm is implemented. Only three of the most efficient ones with the proposed formulation are presented here, for the sake of brevity. They are executed repetitively as new columns are generated during the branch-and-price algorithm, with the aim of constructing integer solutions using the pool of columns available at the current node.

Simple rounding. Given a feasible solution to the linear relaxation of the set-cover formulation, simple rounding rounds up each variable to obtain a feasible integer solution.

One-opt. One-opt starts from a feasible solution to the set-cover formulation, for example provided by Simple Rounding, and tries to improve it, by shifting some values of the variables from 1 to 0 whenever possible.

ZI Round. Given a feasible solution λ to the linear relaxation of the set-cover formulation, ZI Round defines the fractionality for each variable λ_B as $ZI(\lambda_B) = \min\{\lambda_B - \lfloor\lambda_B\rfloor, \lceil\lambda_B\rceil - \lambda_B\}$ and the fractionality of the solution as $ZI(\lambda) = \sum_{B\in\mathcal{B}} ZI(\lambda_B)$. The goal of ZI Round is to identify the variables that can be rounded to improve $ZI(\lambda)$ until the integer infeasibility becomes 0 at which point an integral solution has been found. Therefore, each variable λ_B is rounded in the direction

that reduces $ZI(\lambda_B)$ the most. In the case of a tie, the variable is rounded down. The ZI Round heuristic is described in detail by Wallace (2010).

4. Numerical experiments

We conduct, with three main purposes, numerical experiments based on standard instances of BP, from which we construct instances with uncertain-size items. First, we explore numerically the tractability of these problems by the proposed branch-and-price algorithm. Second, we estimate numerically the costs of robustness, in terms of the average additional numbers of required bins, for the three proposed robust approaches. Third, we perform simulations of solution feasibility under various scenarios generated with different probability distributions for increasing variations of uncertain sizes. Last, an evaluation of the ability of the proposed branch-and-price algorithm to deal with BP, using as a reference results found in the literature, is provided in Appendix B.

The branch-and-price algorithm is implemented in C on top of the SCIP Optimization Suite 7.0.2. IBM CPLEX 12.8 is used as the linear programming solver. All numerical experiments were performed on an Intel Core i7-6700HQ CPU at 2.6 GHz with 4 GB of RAM. The time limit was set to ten minutes of running time per instance. For the branch-and-bound algorithm which solves the knapsack problem with conflicts, the limit on the number of nodes, active only when an improving column has been found, is set to $10000|V|$. Depth first search is restarted from the best-bound node every 100 nodes. Average results are reported here, but full results are available at: `https://tinyurl.com/y6myu7uo`.

4.1. Adaptation of the BP instances to $RBP_1(r)$, $RBP_\infty(r)$ and $RBP_{rr}(\alpha)$

We considered all the most used BP benchmark sets from the literature, and selected those where item sizes are sufficiently lower than bin capacity, to be turned into feasible robust instances. BP instances from the following studies were adapted to $RBP_1(r)$, $RBP_\infty(r)$ and $RBP_{rr}(\alpha)$.

Falkenauer (1996). We use the 80 instances from the set *Falkenauer T*, which have between 60 and 501 items, a bin capacity equal to 1000, and item sizes in the range $\{250, \cdots, 499\}$.

Scholl et al. (1997). We use the 10 instances from the set *Scholl 3*. Their number of items is uniformly distributed between 50 and 500. The bin capacity is equal to 100,000. Item sizes are in the range $\{20000, \cdots, 35000\}$.

Schwerin and Wäscher (1997). We use two sets of 100 instances, one with 100 items, *Schwerin 1*, the other with 120 items, *Schwerin 2*, and a bin capacity equal to 10,000. Item sizes are in the range $\{150, \cdots, 200\}$.

Let P stand for the probability for an item to have an uncertain size. We set $P = 0.1$ to generate 290 instances, in which a binary flag is simply added to indicate whether an item size is uncertain or not, then $P = 0.3$ to generate 290 others and finally $P = 0.5$ to generate the last 290. In the following, these instances are used for non-robust BP, treating for this problem uncertain sizes as certain, as well as for the robust approaches $RBP_1(r)$, $RBP_\infty(r)$ and $RBP_{rr}(\alpha)$, for which r or α is considered as a separate parameter.

4.2. Results of the proposed branch-and-price algorithm for the robust variants of BP

In Tables 3, 4 and 5, we report aggregated results obtained by the proposed branch-and-price algorithm for $RBP_1(r)$, $RBP_\infty(r)$ and $RBP_{rr}(\alpha)$, respectively. Results are grouped by percentage of uncertain-size items and by value of r or α. Each row provides aggregated results with 290 instances.

Column "$\widetilde{V}, (\%)$" indicates the percentage of uncertain-size items. Column "r" or "α" provides the minimum value of the robustness indicator. The four other columns give aggregated values for the corresponding group of instances. Column "*Min.* `CPU`*, (s.)*" gives the minimum solution time in seconds. Column "*Avg.* `CPU`*, (s.)*" gives the average solution time in seconds. Column "*Max.* `CPU`*, (s.)*" gives the maximum solution time in seconds. Column "*Std.* `CPU`*, (s.)*" gives the standard deviation of solution time in seconds. Column "*Avg.* #`BIN`" indicates the average number of bins in the best found solutions. Column "#`OPT`, (%)" provides the percentage of proven optimal solutions obtained. Column "#`SOL`, (%) *with* `GAP` $= 1$" gives the percentage of solutions with an absolute gap of one with the best known lower bound at the end of the optimization process.

In every case, the proposed algorithm either converged to a proven optimal solution or obtained a solution with an absolute gap of one to the best known lower bound. Hence, for each row, the sum of the two last columns equals 100 percent. As the proposed algorithm always obtains an optimal solution with $RBP_\infty(r)$ and $RBB_{rr}(\alpha)$, the last column is omitted in Tables 4 and 5.

4.2.1. Results of the proposed branch-and-price algorithm with $RBP_1(r)$

In Table 3, we report average results of the proposed branch-and-price algorithm with $RBP_1(r)$, aggregated by percentage of uncertain-size items and by value of r.

$\widetilde{V}$, (%)	r	Min. CPU, (s.)	Avg. CPU, (s.)	Max. CPU, (s.)	Std. CPU, (s.)	Avg. #BIN	#OPT, (%)	#SOL, (%) with GAP = 1
10%	All	0.09	52.28	616.78	161.45	39.69	92.3%	7.7%
	0.2W (290)	0.11	38.46	605.82	137.26	38.52	94.83%	5.17%
	0.3W (290)	0.1	40.26	616.78	141.86	39.44	94.14%	5.86%
	0.4W (290)	0.09	78.1	603.47	195.99	41.1	87.93%	12.07%
30%	All	0.03	55.19	601.78	168.88	44.2	91.26%	8.74%
	0.2W (290)	0.06	63.11	601.78	179.58	40.64	90%	10%
	0.3W (290)	0.08	38.08	600.21	140.7	43.71	94.14%	5.86%
	0.4W (290)	0.03	64.37	600.21	182.4	48.26	89.66%	10.34%
50%	All	0.02	64.59	600.88	182.2	48.63	89.66%	10.34%
	0.2W (290)	0.05	65.43	600.02	182.21	42.68	89.66%	10.34%
	0.3W (290)	0.02	62.53	600.88	179.7	47.76	90%	10%
	0.4W (290)	0.02	65.8	600.01	185.27	55.44	89.31%	10.69%
All	All	0.02	57.35	616.78	171.07	44.17	91.07%	8.93%

Table 3: Results of the proposed branch-and-price algorithm with $RBP_1(r)$, 870 runs

With $RBP_1(r)$, the proposed branch-and-price algorithm obtains an optimal solution 91% of the times. Its average running time is less than one minute, and its running time is greater than one minute for 37% of the runs.

At the end of the optimization process, the absolute gap between the rounded-up lower bound and the best-found integer feasible solution value, within the time limit of ten minutes, is always lower than or equal to one. This means that either one extra bin is used, compared to an optimal solution, or that the solution is optimal, but the rounded-up lower bound is one unit below. In the latter case, proving optimality requires branching to increase the lower bound, which may be computationally expensive, and generally can't be achieved within the time limit of 600 seconds. Only nine such instances were solved within this time limit.

4.2.2. Results of the proposed branch-and-price algorithm with $RBP_\infty(r)$

In Table 4, we report average results of the proposed branch-and-price algorithm with $RBP_\infty(r)$, aggregated by percentage of uncertain-size items and by value of r.

$\widetilde{V}$, (%)	r	Min. CPU, (s.)	Avg. CPU, (s.)	Max. CPU, (s.)	Std. CPU, (s.)	Avg. #BIN	#OPT, (%)
10%	All	0.04	2.47	56.2	4.68	42.04	100%
	0.2W (290)	0.11	2.65	21.74	4.7	40.37	100%
	0.3W (290)	0.04	2.37	55.7	4.15	41.81	100%
	0.4W (290)	0.06	2.4	56.2	5.15	43.94	100%
30%	All	0	1.54	180.43	8.47	52.25	100%
	0.2W (290)	0.05	1.65	20.73	3.06	46.62	100%
	0.3W (290)	0.02	2.47	180.43	14.26	51.83	100%
	0.4W (290)	0	0.48	4.99	1.06	58.31	100%
50%	All	0	0.37	16.59	0.93	65.3	100%
	0.2W (290)	0.01	0.77	16.59	1.51	53.1	100%
	0.3W (290)	0.01	0.31	1.08	0.19	63.04	100%
	0.4W (290)	0	0.04	1.43	0.11	79.77	100%
All	All	0	1.46	180.43	5.68	53.2	100%

Table 4: Results of the proposed branch-and-price algorithm $RBP_{\infty}(r)$, 870 runs

With $RBP_{\infty}(r)$, the proposed branch-and-price algorithm always obtains an optimal solution, in less than 2 seconds on average. The algorithm requires only six times more than 30 seconds to converge, with a maximum solving time at 180 seconds. The gap between the rounded-up value of linear relaxation at root node and the optimal integer solution value is always equal to zero.

4.2.3. Results of the proposed branch-and-price algorithm with $RBP_{rr}(\alpha)$

In Table 5, we report average results of the proposed branch-and-price algorithm with $RBP_{rr}(\alpha)$, aggregated by percentage of uncertain-size items and by value of α.

$\widetilde{V}$, (%)	α	Min. CPU, (s.)	Avg. CPU, (s.)	Max. CPU, (s.)	Std. CPU, (s.)	Avg. #BIN	#OPT, (%)
10%	All	0.09	3.01	25.23	4.91	38.2	100%
	0.2W (290)	0.13	2.71	22.84	4.49	37.88	100%
	0.3W (290)	0.09	3.15	20.97	5.04	38.13	100%
	0.4W (290)	0.13	3.16	25.23	5.17	38.57	100%
30%	All	0.03	2.78	28.34	4.63	40.45	100%
	0.2W (290)	0.1	2.92	28.34	4.89	39.38	100%
	0.3W (290)	0.03	2.87	27.33	4.85	40.44	100%
	0.4W (290)	0.07	2.55	19.36	4.09	41.52	100%
50%	All	0.02	2.17	21.32	3.26	42.68	100%
	0.2W (290)	0.06	2.53	21.32	3.96	40.83	100%
	0.3W (290)	0.04	2.25	15.89	3.36	42.61	100%
	0.4W (290)	0.02	1.72	11.67	2.17	44.61	100%
All	All	0.02	2.65	28.34	4.34	40.44	100%

Table 5: Results of the proposed branch-and-price algorithm with $RBP_{rr}(\alpha)$, 870 runs

With $RBP_{rr}(\alpha)$, the proposed branch-and-price algorithm always obtains an optimal solution,

in less than 3 seconds on average. The algorithm always requires less than 30 seconds to converge. The gap between the rounded-up value of linear relaxation at root node and the optimal integer solution value is always equal to zero.

4.3. Numerical evaluation of the extra cost for robustness

In the following, the extra cost for robustness is measured as the percentage of extra-bins required in the best found robust solution, for the considered robust approach, compared to the best found non-robust BP solution, to the same base instance. Concerning the BP solutions, 868 out of 870 are optimal, and the two others may be optimal, or use at most one extra bin.

4.3.1. Extra costs for robustness with $RBP_1(r)$ and $RBP_\infty(r)$

In Figure 1, we report the increase of the number of bins in percentages, compared to the non-robust case, for $RBP_1(r)$ and $RBP_\infty(r)$, with a percentage of uncertain-size items varying from 10%, 30% to 50%, and r varying by $0.2W$, $0.3W$ and $0.4W$.

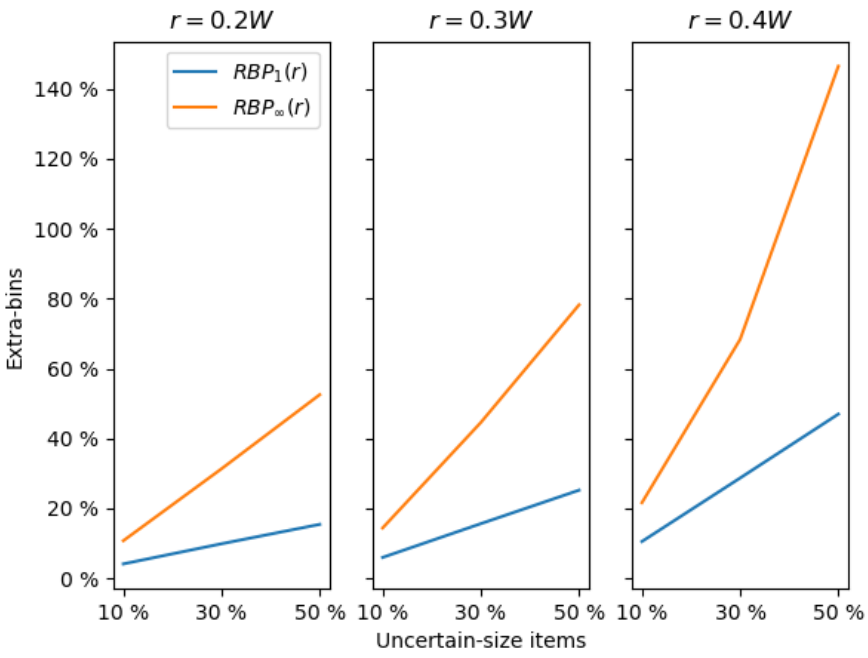

Figure 1: Increase of the extra cost for robustness, varying the percentage of uncertain-size items and r, for $RBP_1(r)$ and $RBP_\infty(r)$

The extra cost for robustness ranges from 4% to 146% of extra bins, depending on the robust approach, the percentage of uncertain-size items, and r. For $RBP_1(r)$, this extra cost remains moderate for $\widetilde{V}, (\%) = 10\%$, ranging from 4% to 11% of extra bins. It grows up to 47% of extra bins

for the maximum values of $r = 0.4W$ and $\widetilde{V},(\%) = 50\%$. For $RBP_\infty(r)$, this cost sharply increases with $\widetilde{V},(\%)$, particularly for larger value of r: for $r = 0.4W$, it ranges from 22% to 146%, as $\widetilde{V},(\%)$ increases.

Besides, we observe that the increase of the number of bins, compared to the non-robust case, is on average 2.8 times greater with $RBP_\infty(r)$ than with $RBP_1(r)$. As can be easily proven, with the same value of the stability radius, and the same uncertain items, the extra cost with the ℓ_∞ norm is always greater than or equal to the one with the ℓ_1 norm.

4.3.2. Extra cost for robustness with $RBP_{rr}(\alpha)$

In Table 6, we report the increase of the number of bins in percentages, compared to the non-robust case, for $RBP_{rr}(\alpha)$, with a percentage of uncertain-size items equal to 10%, 30% or 50%, when α is set to 0.2 or 0.3 or 0.4.

Column "$\widetilde{V},(\%)$" indicates the percent of uncertain-size items. Column "α" indicates the minimum relative resiliency. Column *"Extra bins, (%)"* provides the percent of extra bins required compared to the non-robust case.

α	$\widetilde{V},(\%)$	Extra bins, (%)
0.2	10%	1.97%
	30%	5.80%
	50%	9.69%
0.3	10%	2.48%
	30%	8.56%
	50%	14.34%
0.4	10%	3.68%
	30%	11.54%
	50%	19.47%

Table 6: Extra cost for robustness increasing relative resiliency, for $RPB_{rr}(\alpha)$

With relative resiliency, for 10% of uncertain items, the extra cost for robustness is almost negligible, under 4% of extra bins, even for $\alpha = 0.4$. It remains under 10% for $\alpha \leq 0.3$, excepted for $\alpha = 0.3$ and $\widetilde{V},(\%) = 50\%$. It reaches a maximum of 19% for the maximum considered values of α and $\widetilde{V},(\%)$. In the next section, we investigate through simulation the solution feasibility under various scenarios, for the proposed robust approaches, compared to a non-robust one.

4.4. Simulation of solution feasibility under various scenarios

We perform numerical simulation of solution feasibility under various scenarios. Random variables describing uncertain-size variations are assumed to be independent and identically distributed, with $[-1, 1]$ as probability distribution's support. Five symmetric probability distributions are considered: truncated normal, triangular, bimodal, uniform and beta(0.5, 0.5). A variation is obtained by multiplying the draw of a random variable by a level of variation and by either bin capacity or current item size. Negative variations are converted to 0. Ten levels of variations are tested, from 0.05 to 0.5, by increment of 0.05.

For each of the 870 instances described in Section 4.1, each of the 10 levels of variation, each of the 5 probability distributions, in the case of variations relative to bin capacity and in the case of variations relative to item size, 1 000 scenarios are generated, resulting in a total of 87 000 000 scenarios. The feasibility of each solution obtained by the considered approaches, with the considered values for parameters r or α, is evaluated under each of the 100 000 scenarios generated for the corresponding instance.

For each probability distribution, in the case of variations relative to bin capacity and in the case of variations relative to item size, we report in Figure 2 the overall feasibility probability after real item sizes are known for each considered approach, against an increasing level of variation of uncertain sizes.

For a given probability distribution and a given level of variation, this overall feasibility probability of an approach, for a given value of r or α, is estimated as the number of scenarios that have a feasible solution, over the total number of scenarios.

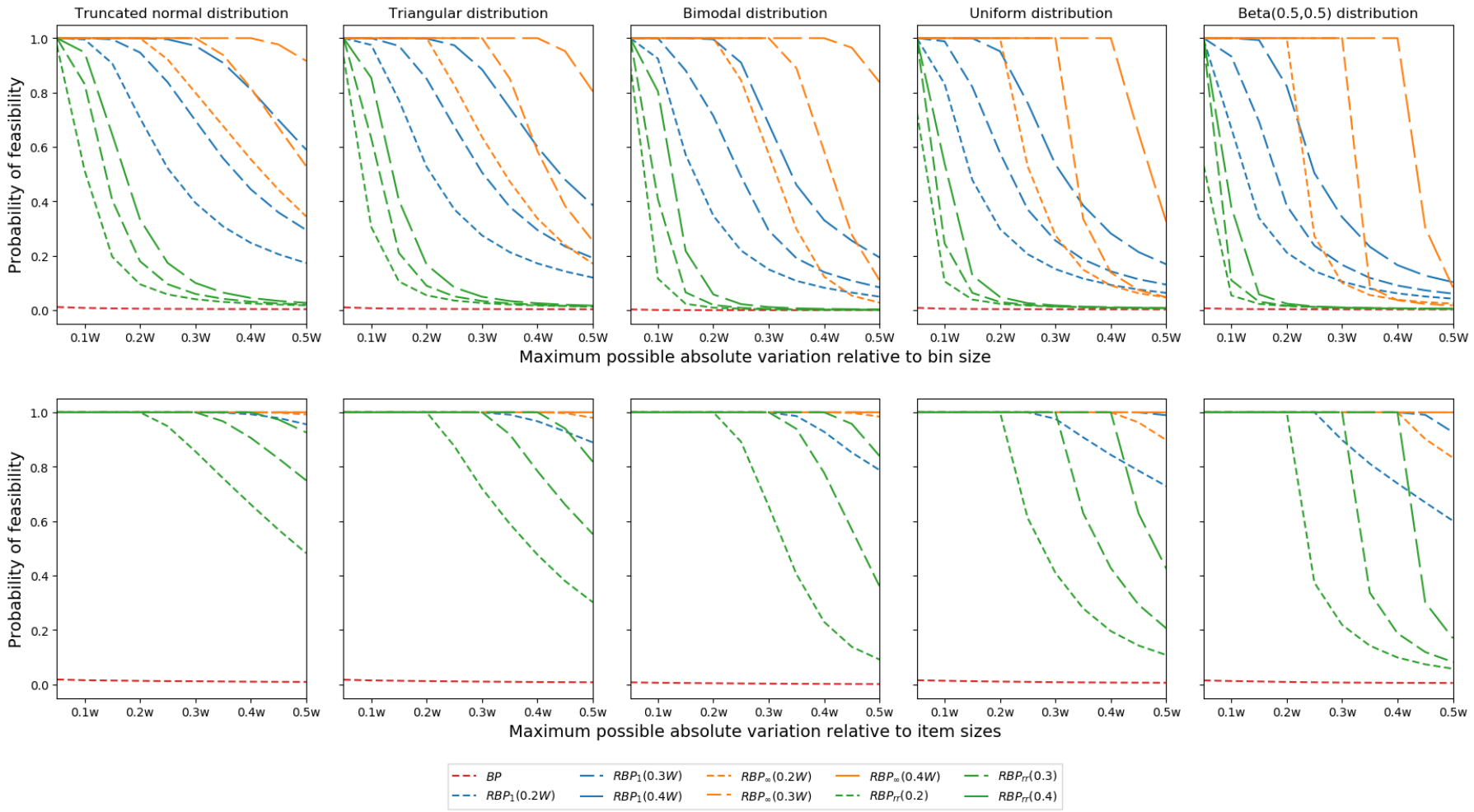

Figure 2: Simulation results of solution feasibility under five probability distributions, increasing maximum possible absolute variation, relative to bin size or item sizes

We observe that standard BP provides solutions with a feasibility probability always close to 0, as they don't reserve any free space in the used bins, that could absorb positive variations, and as they use as few bins as possible. Hence, in the case of uncertain sizes, if feasibility is required, a robust approach is a necessity.

Considering variations relative to bin capacity, therefore unrelated to each nominal size, only solutions to $RBP_\infty(r)$ have a guarantee of feasibility, as long as any variation is not greater than r. Although the cost of robustness with $RBP_\infty(r)$ is high (see Section 4.3.1), it is a requirement for this guarantee, according to Property 4. Once r is exceeded, the feasibility probabilities always quickly decrease, as r increases, at a rate that depends on the probability distribution, and which is higher when more extreme values are generated, for example with the beta(0.5, 0.5) distribution. Hence, these solutions are not overly conservative, even in the presence of negative variations, as in these simulations, which compensate positive ones.

A solution to $RBP_1(r)$ comes with some protection against uncertainty, but generally no guarantee of feasibility. Its feasibility probability is close to one for a range of variation level that depends on the probability distribution and on r. With the truncated normal distribution, this probability is greater than 0.7 with a variation level not greater than r. For probability distributions generating more extreme values, this probability is lower.

As can be expected, relative resiliency based approaches are inadequate for variations that are not related to nominal item sizes, as they have a low probability of feasibility.

Finally, in the case of variations related to nominal sizes, relative resiliency is a more adequate approach, which offers both a feasibility guarantee and a smaller cost for robustness. In this case, the proposed stability radius based approaches require more bins to offer the same protection against uncertainty.

5. Conclusion and perspectives

In this paper, we introduced three robust variants of the BP problem with items of uncertain size. After the assignment of items to bins has been decided, the size of these items can vary. The objective was to minimize the number of bins used, while ensuring the chosen level of robustness. Three indicators of robustness were studied: the stability radius calculated by using the ℓ_1 norm, the stability radius calculated by using the ℓ_∞ norm, and the relative resiliency, in which possible variations are related to nominal size values. A unique 0-1 linear program was designed, valid for the three robust variants of the BP problem. A set-cover reformulation of the model was done with an exponential number of columns, but a stronger linear relaxation bound. A branch-and-price

algorithm with dynamic column generation was developed. Then, a thorough numerical study was performed with thousands of standard benchmark BP instances, in which randomly chosen item sizes were set are uncertain.

We observed that the proposed algorithm is able to obtain an optimal solution to every instance of $RBP_\infty(r)$ and $RBP_{rr}(\alpha)$, on average in less than three seconds per instance. It also provided a proven optimal solution to 91% of $RBP_1(r)$ instances, and a solution either optimal or requiring one extra bin to the remaining ones, on average in less than one minute.

We also measured the extra cost for robustness, which ranged from 2% to 146% extra bins, depending on the robust approach, the percentage of uncertain items, and the value of r or α. Then, we performed simulation of solution feasibility under various scenarios, generated with different probability distributions for increasing variations. We observed that standard BP provides solutions with a feasibility probability always close to 0. In the case of variations relative to bin capacity, for all the considered robust approaches, feasibility probability always decreases relatively fast, as the level of variation increases. The solutions obtained with $RBP_\infty(r)$ have a greater cost for robustness but come with a guarantee of feasibility when any variation is not greater than r. With variations observing a truncated normal distribution, a solution to $RBP_1(r)$ had a feasibility probability greater than 0.7 with a variation level not greater than r. For probability distributions generating more extreme values, this bound was lower. Finally, in the case of variations relative to item sizes, $RBP_{rr}(\alpha)$ offers both a feasibility guarantee and a smaller cost for robustness.

This work offers several perspectives. First, one could consider different robustness models, with for example uncertain bins, in which all item sizes can deviate, or with different kinds of uncertainties, affecting different subsets of items. Second, the proposed branch-and-price algorithm could be improved. To achieve this, a possible research direction is related to the acceleration of the lower bound evaluation, another one to the improvement of this lower bound, and another one to the design of primal heuristics.

Appendix A Formal definitions

A.1 Definition of stability radius ρ_1, stability radius ρ_∞ and relative resiliency ρ_{rr}

To evaluate the robustness of a feasible solution, we use the stability radius concept, whose formal definition requires some additional notations to the ones provided in Section 3.1:

- $w = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n_+$, a vector expressing nominal item sizes. For any item $i \in V$ of certain size, nominal and real sizes are assumed to be equal. For any item $i \in \widetilde{V}$ of uncertain size, there may be a difference between nominal and real sizes,

- $\Xi = \{\xi \in \mathbb{R}^n \mid \xi_i = 0,\ i \in V \setminus \widetilde{V}\}$, a set of vectors, such that each vector $\xi \in \Xi$ represents a possible scenario of uncertain-size variations,
- $F(w)$, the set of feasible solutions with respect to a given vector $w \in \mathbb{R}^n_+$,
- $\widetilde{K} = \{k \in K \mid V^k \cap \widetilde{V} \neq \emptyset\}$, the set of bins of a given feasible solution having at least one uncertain item, where V^k is the set of items assigned to bin k.

The stability radius of a feasible solution $s \in F(w)$ can then be defined as follows (see Sotskov et al. (2006)): $\rho(s, w) = \max\{\epsilon \geq 0 \mid \forall \xi \in B(\epsilon),\ s \in F(w + \xi)\}$, where $B(\epsilon) = \{\xi \in \Xi \mid \|\xi\| \leq \epsilon\}$.

Hence, $\rho(s, w)$ is determined as the value of the radius of the greatest closed ball $B(\cdot)$, called the stability ball which represents uncertain size variations, for which s remains feasible. Any element ξ of $B(\cdot)$ is evaluated based on a given norm $\|\cdot\|$ defining the distance between vectors w and $w + \xi$ (or the amplitude of variations from w). In this paper, two norms ℓ_1 ($\|\cdot\|_1$) and ℓ_∞ ($\|\cdot\|_\infty$) are considered, where by definition $\|\xi\|_1 = \sum_{i \in \widetilde{V}} |\xi_i|$ and $\|\xi\|_\infty = \max\limits_{i \in \widetilde{V}}\{|\xi_i|\}$. As a consequence, the notations ρ_1 and ρ_∞ will be used for ℓ_1 and ℓ_∞, respectively.

The integer programming formulation of RBP_1, RBP_∞ and RBP_{rr} is based on the following three theorems, which are demonstrated by Rossi et al. (2016), by Sotskov et al. (2006) and by Pirogov (2019), respectively.

Theorem 1 (Rossi et al. (2016))**.** *The stability radius ρ for a given feasible solution is calculated as follows:* $\rho_1 = \min\limits_{k \in \widetilde{K}} \left\{W - \sum_{i \in V^k} w_i\right\}$.

Theorem 2 (Sotskov et al. (2006))**.** *The stability radius ρ_∞ for a given feasible solution is calculated as follows:* $\rho_\infty = \min\limits_{k \in \widetilde{K}} \left\{\frac{W - \sum_{i \in V^k} w_i}{|V^k \cap \widetilde{V}|}\right\}$.

For some practical applications, a possible limitation of theses approaches is that variations are not related to nominal size values. Therefore, we consider another robust approach, proposed by Pirogov (2019), based on the relative resiliency.

The relative resiliency $\rho_{rr}(s, w)$ refers to the situation where each uncertain-size variation is bounded by an amount that is proportional to the nominal value defined by a coefficient $\alpha > 0$. The set of vectors representing possible scenarios of uncertain-size variations is then defined as $\Xi(\alpha, w) = \{\xi \in \Xi \mid \xi_i \leq \alpha w_i,\ i \in \widetilde{V}\}$. In this case, the relative resiliency $\rho_{rr}(s, w) = \max\{\alpha \geq 0 \mid s \in F(w + \xi),\ \xi \in \Xi(\alpha, w)\}$ is the greatest value of α for which the solution s remains feasible.

Theorem 3 (Pirogov (2019))**.** *The relative resiliency ρ_{rr} for a given feasible solution is calculated as follows:* $\rho_{rr} = \min\limits_{k \in \widetilde{K}} \left\{\frac{W - \sum_{i \in V^k} w_i}{\sum_{i \in V^k \cap \widetilde{V}} w_i}\right\}$.

A.2 Robust bin-packing problems $RBP_1(r)$, $RBP_\infty(r)$ and $RBP_{rr}(\alpha)$

We define:

- $RBP_1(r)$ (resp. $RBP_\infty(r)$) as the problem of obtaining a solution to BP with a stability radius in norm ℓ_1 (resp. ℓ_∞) at least equal to $r \in \mathbb{R}_+$ using a minimum number of bins.
- $RBP_{rr}(\alpha)$ as the problem of obtaining a solution to BP with a relative resiliency at least equal to $\alpha \in \mathbb{R}_+$ using a minimum number of bins.

We introduce the following property which states that adding a value $r \in \mathbb{R}_+$ to each uncertain size ensures that any feasible solution to the resulting instance of BP has a stability radius ρ_∞ at least equal to r in the original instance.

Property 4. *A given feasible solution of BP has the stability radius ρ_∞ at least equal to r iff the following inequality $\sum_{i \in V^k \cap \widetilde{V}} (w_i + r) + \sum_{i \in V^k \setminus \widetilde{V}} w_i \leq W$ holds for any bin $k \in \widetilde{K}$.*

Proof. The latter inequality is equivalent to the following one $\sum_{i \in V^k} w_i + |V^k \cap \widetilde{V}| \cdot r \leq W$, which can be transformed into $r \leq \frac{W - \sum_{i \in V^k} w_i}{|V^k \cap \widetilde{V}|}$ and, as a consequence, we have $r \leq \min_{k \in \widetilde{K}} \left\{ \frac{W - \sum_{i \in V^k} w_i}{|V^k \cap \widetilde{V}|} \right\}$. Based on Theorem 2, we finally obtain that $r \leq \rho_\infty$. □

In Figure A.3, a robust solution to an instance of BP is presented with a total of five items, three uncertain sizes (items 2, 3 and 5), a bin capacity of ten, and a stability radius ρ_∞ equal to one. Property 4 states that to obtain such a solution, with a stability radius ρ_∞ at least equal to one, it is possible to add one to each uncertain size, and to solve the resulting BP instance, as it appears in the figure.

Similarly, the following property states that adding a value αw_i to each uncertain size w_i for $i \in \widetilde{V}$ ensures that any feasible solution to the resulting instance of BP will have a relative resiliency ρ_{rr} at least equal to α in the original instance.

Property 5. *A given feasible solution of BP has a relative resiliency ρ_{rr} at least equal to α iff the following inequality $\sum_{i \in V^k \cap \widetilde{V}} (1 + \alpha) w_i + \sum_{i \in V^k \setminus \widetilde{V}} w_i \leq W$ holds for any bin $k \in \widetilde{K}$.*

Proof. The latter inequality is equivalent to the following one $\alpha \cdot \sum_{i \in V^k \cap \widetilde{V}} w_i + \sum_{i \in V^k \cap \widetilde{V}} w_i \leq W - \sum_{i \in V^k \setminus \widetilde{V}} w_i$, which can be transformed into $\alpha \leq \frac{W - \sum_{i \in V^k} w_i}{\sum_{i \in V^k \cap \widetilde{V}} w_i}$ and, as a consequence, we have $\alpha \leq \min_{k \in \widetilde{K}} \left\{ \frac{W - \sum_{i \in V^k} w_i}{\sum_{i \in V^k \cap \widetilde{V}} w_i} \right\}$. Based on Theorem 3, we finally obtain that $\alpha \leq \rho_{rr}$. □

Property 4 (resp. Property 5) implies that a solution to any instance of $RBP_\infty(r)$ (resp. $RBP_{rr}(\alpha)$) can be obtained by solving an instance of BP, in which item sizes are chosen accordingly.

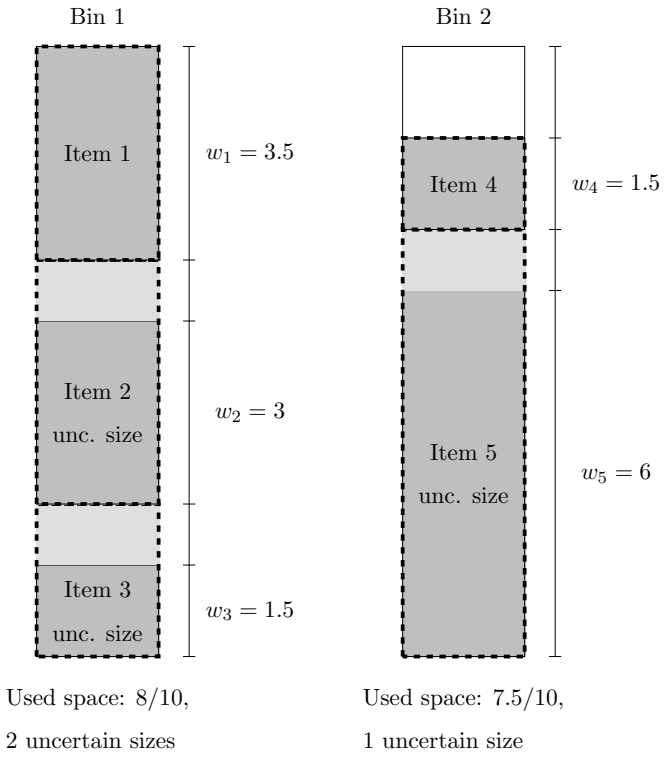

Figure A.3: A robust solution to an instance of BP with $\rho_\infty = 1$, obtained by reserving one unit of free space for each uncertain size

However, for $RBP_1(r)$, we can't use the same kind of straightforward reduction, as implied by the following Property.

Property 6. *A given feasible solution of BP has a stability radius ρ_1 at least equal to r iff the following inequality $\sum_{i \in V^k} w_i + r \leq W$ holds for any bin $k \in \widetilde{K}$.*

Proof. The latter inequality is equivalent to the following one $r \leq W - \sum_{i \in V^k} w_i$, which conducts us to $r \leq \min_{k \in \widetilde{K}} \left\{ W - \sum_{i \in V^k} w_i \right\}$. Based on Theorem 1, we finally obtain that $r \leq \rho_1$. □

Hence, with $RBP_1(r)$, any bin containing one uncertain item or more must keep a quantity of unused space at least equal to r, which led us to the 0-1 linear programming formulation of Section 3.1.

Appendix B Numerical results with BP

We provide numerical results for BP, with instances from the benchmark sets proposed by Falkenauer (1996), Scholl et al. (1997), Wäscher and Gau (1996), Schwerin and Wäscher (1997), Schoenfield (2002) and Delorme et al. (2016). We restrict the report to the subset of 3877 instances

(out of 5450) used by Delorme et al. (2016). The maximum amount of time per instance is set to one minute.

In Table B.7, we report comparative numerical results for BP obtained by the proposed branch-and-price algorithm and other branch-and-price algorithms from the literature. In the column "*Set*", the name of the set of instances is provided. In the column "#INSTANCE", the number of tested instances for the corresponding set of instances is given. In the columns "#OPT", "#OPT Vance", "#OPT SCIP", "#OPT Belov", and "#OPT Song", the number of instances solved to optimality in less than one minute per instance, by the proposed branch-and-price algorithm, and by the ones by Vance et al. (1994), by Gamrath et al. (2016), by Belov and Scheithauer (2006), and by Song et al. (2018), are provided, respectively. For "#OPT Vance", "#OPT SCIP", and "#OPT Belov", the reported results come from the study by Delorme et al. (2016), in which a benchmark of several algorithms for BP was performed on the same computer, an Intel Xeon CPU at 3.1 GHz, with 8 GB of RAM, faster than the Intel Core i7-6700HQ CPU at 2.6 GHz with 4 GB of RAM that we used. An asterisk in a cell indicates that the algorithm obtained the maximum number of optimal solutions. In Song et al. (2018), results are provided for only six instance sets, and a hyphen "-" in a cell indicates that no results are available for the algorithm.

Set	#INSTANCE	#OPT	#OPT Vance	#OPT SCIP	#OPT Belov	#OPT Song
Falkenauer U	74	60	53	18	74*	-
Falkenauer T	80	79*	76	35	57	52
Scholl 1	323	323*	323*	244	323*	-
Scholl 2	244	197	204	67	244*	-
Scholl 3	10	10*	10*	0	10*	0
Wascher	17	15	6	0	17*	6
Schwerin 1	100	100*	100*	0	100*	100*
Schwerin 2	100	99	100*	0	100*	100*
Hard 28	28	26	11	7	28*	11
Random 50	165	165*	165*	165*	165*	-
Random 100	271	271*	271*	271*	271*	-
Random 200	359	359*	358	293	359*	-
Random 300	393	393*	387	155	393*	-
Random 400	425	425*	416	114	425*	-
Random 500	414	412	394	69	414*	-
Random 750	433	407	99	22	433*	-
Random 1000	441	282	62	0	441*	-
Total	3877	3623	3035	1460	3854*	-

Table B.7: Comparative results with BP: number of instances solved in at most one minute per instance

Globally, the proposed algorithm is able to solve 93 percent of the instances of the considered subset in less than one minute per instance. It solves more instances than the previous ones based on the same branching rule: 19 percent more instances than the best previous one proposed by Vance et al. (1994). It solves only 6 percent less instances than the best known branch-and-price algorithm for BP proposed by Belov and Scheithauer (2006).

Regarding the instances which are not solved within one minute, there are two main difficulties. First, in the case of a large number of items or a large bin capacity, solving the linear relaxation of the master problem at root node may require more than one minute. Improving the performance of the proposed branch-and-price algorithm would require accelerating the lower bound evaluation. Second, obtaining an integer optimal solution may require too much branching and column generation. A very efficient tailored heuristic such as the one used by Belov and Scheithauer (2006) and introduced

by Mukhacheva et al. (2000), may be adapted to RBP_1 and incorporated into the algorithm, to speed-up the solving process.

References

Achterberg, T., Berthold, T., and Hendel, G. (2012). Rounding and propagation heuristics for mixed integer programming. In Klatte, D., Lüthi, H.-J., and Schmedders, K., editors, *Operations Research Proceedings 2011*, pages 71–76, Berlin, Heidelberg. Springer Berlin Heidelberg.

Alem, D. J., Munari, P. A., Arenales, M. N., and Ferreira, P. A. V. (2010). On the cutting stock problem under stochastic demand. *Annals of Operations Research*, 179(1):169–186.

Alfandari, L., Plateau, A., and Schepler, X. (2015). A branch-and-price-and-cut approach for sustainable crop rotation planning. *European Journal of Operational Research*, 241(3):872–879.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

Battaïa, O. and Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142(2):259–277.

Belov, G. and Scheithauer, G. (2006). A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European Journal of Operational Research*, 171(1):85–106.

Berthold, T. (2008). Heuristics of the branch-cut-and-price-framework SCIP. In Kalcsics, J. and Nickel, S., editors, *Operations Research Proceedings 2007*, pages 31–36, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53.

Boysen, N., Fliedner, M., and Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, 183(2):674–693.

Cardoen, B., Demeulemeester, E., and Beliën, J. (2010). Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932.

Crainic, T. G., Gobbato, L., Perboli, G., and Rei, W. (2016). Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic. *European Journal of Operational Research*, 253(2):404–417.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20.

Desrosiers, J. and Lübbecke, M. (2011). Branch-price-and-cut algorithms. In Cochran, J., editor, *Encyclopedia of Operations Research and Management Science*. John Wiley & Sons.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2(1):5–30.

Farley, A. A. (1990). A note on bounding a class of linear programming problems, including cutting stock problems. *Operations Research*, 38(5):922–923.

Fukasawa, R., Longo, H., Lysgaard, J., De Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511.

Gamrath, G. (2010). *Generic branch-cut-and-price*. PhD thesis, Technischen Universität Berlin.

Gamrath, G., Fischer, T., Gally, T., Gleixner, A. M., Hendel, G., Koch, T., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M. E., et al. (2016). The SCIP Optimization Suite 3.2. Technical report, Zuse Institute Berlin, Germany.

Horowitz, E. and Sahni, S. (1974). Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21(2):277–292.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack problems*. Springer, Berlin.

Lamiri, M., Xie, X., Dolgui, A., and Grimaud, F. (2008). A stochastic model for operating room planning with elective and emergency demand for surgery. *European Journal of Operational Research*, 185(3):1026–1037.

Lemaréchal, C. (2007). The omnipresence of lagrange. *Annals of Operations Research*, 153(1):9–27.

Martello, S., Pisinger, D., and Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424.

Mukhacheva, E., Belov, G., Kartack, V., and Mukhacheva, A. (2000). Linear one-dimensional cutting-packing problems: numerical experiments with the sequential value correction method (SVC) and a modified branch-and-bound method (MBB). *Pesquisa Operacional*, 20(2):153–168.

Pirogov, A. (2019). *Robust balancing of production lines: MILP models and pre-processing rules*. PhD thesis, IMT Atlantique, Nantes, France.

Rossi, A., Gurevsky, E., Battaïa, O., and Dolgui, A. (2016). Maximizing the robustness for simple assembly lines with fixed cycle time and limited number of workstations. *Discrete Applied Mathematics*, 208:123–136.

Ryan, D. and Foster, E. (1981). An integer programming approach to scheduling. *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, page 269–280.

Sadykov, R. and Vanderbeck, F. (2013). Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255.

Scheithauer, G. and Terno, J. (1997). Theoretical investigations on the modified integer round-up property for the one-dimensional cutting stock problem. *Operations Research Letters*, 20(2):93–100.

Schoenfield, J. E. (2002). Fast, exact solution of open bin packing problems without linear programming. Technical report, US Army Space and Missile Defense Command, Huntsville, Alabama, USA.

Scholl, A., Klein, R., and Jürgens, C. (1997). Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, 24(7):627–645.

Schwerin, P. and Wäscher, G. (1997). The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, 4(5):377–389.

Song, G., Kowalczyk, D., and Leus, R. (2018). The robust machine availability problem – bin packing under uncertainty. *IISE Transactions*, 50(11):997–1012.

Sotskov, Y. N., Dolgui, A., and Portmann, M.-C. (2006). Stability analysis of an optimal balance for an assembly line with fixed cycle time. *European Journal of Operational Research*, 168(3):783–797.

Toth, P. (1980). Dynamic programming algorithms for the zero-one knapsack problem. *Computing*, 25(1):29–45.

Vance, P. H., Barnhart, C., Johnson, E. L., and Nemhauser, G. L. (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, 3(2):111–130.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128.

Wallace, C. (2010). ZI round, a MIP rounding heuristic. *Journal of Heuristics*, 16(5):715–722.

Wäscher, G. and Gau, T. (1996). Heuristics for the integer one-dimensional cutting stock problem: A computational study. *Operations-Research-Spektrum*, 18(3):131–144.

Yue, M. (1990). A simple proof of the inequality FFD $(L) \leq 11/9$ OPT $(L) + 1$, $\forall L$ for the FFD bin-packing algorithm. *Acta Mathematicae Applicatae Sinica*, 7:321–331.

Östergård, P. R. (2002). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1):197–207.