ON THE ROBUST KNAPSACK PROBLEM

MICHELE MONACI * AND ULRICH PFERSCHY †

Abstract. We consider an uncertain variant of the knapsack problem that arises when the exact weight of each item is not exactly known in advance but belongs to a given interval, and the number of items whose weight differs from the nominal value is bounded by a constant. We analyze the worsening of the optimal solution value with respect to the classical problem, and exactly determine its worst-case performance depending on uncertainty for all parameter configurations. We perform the same analysis for the fractional version of the problem in which one is allowed to pack any fraction of the items. In addition, we derive the worst-case performance ratio, with respect to the optimal solution value, for both the fractional problem and for a variant of the well-known greedy algorithm. Finally, we consider a relevant special case and provide a combinatorial algorithm for solving the fractional problem in an efficient way.

Key words. knapsack problem, robust optimization, worst-case ratio

AMS subject classifications. 90C27, 90C59, 90C90

1. Introduction. One of the most relevant assumptions when solving an optimization problem is that all input data are known in advance. Unfortunately, this is not always the case when real-world problems are considered, since the models we use only provide an approximation of real systems and because uncertainty can change the effective values of some input parameters with respect to nominal values. Two main methods have been proposed in the literature to deal with uncertainty. Stochastic optimization is used when a finite number of scenarios are possible and some information about the probabilistic distribution of effective data is available. On the contrary, Robust Optimization is based on the definition of a *robust counterpart* for a nominal problem, in which hard constraints are imposed to forbid solutions that have a large probability to become infeasible. Within these settings, Soyster [24] defined all those solutions that remain feasible for all input data belonging to a given convex set as robust solutions. Indeed, the quality of these solutions turns out to be quite poor with respect to the optimal solution of the nominal problem. A different definition of robustness was provided by Ben-Tal and Nemirovski [4, 5], who considered the case in which uncertainty belongs to a given ellipsoidal set. The resulting models, despite tractable, turn out to be quite hard to solve in practice. A breakthrough in robust optimization is the definition of robustness given by Bertsimas and Sim [8], which has the attractive aspect that, for a linear problem, robustness can be enforced by solving a robust formulation that is also a linear problem.

All these methods and their variants have been addressed in the literature from a computational viewpoint; in addition, bounds on the probability for the produced solutions to be feasible were given under some uncertainty distributions. However, to the best of our knowledge, no attention has been paid on the effects of robustness on the solution value from a worst-case point of view. In this paper, we perform a worst-case analysis of the solution loss due to robustness, providing an understanding of when robustness is cheap or expensive (see, Bertsimas, Brown and Caramanis [6]).

In particular, we consider a set $N = \{1, \dots, n\}$ of items, each of them with positive profit p_j and positive weight w_j, and a knapsack having capacity c. The *Knapsack*

*DEI, University of Padova, Via Gradenigo 6/A, I-35131 Padova, Italy (monaci@dei.unipd.it).

†Department of Statistics and Operations Research, University of Graz, Universitaetsstrasse 15, A-8010 Graz, Austria (pferschy@uni-graz.at).

Problem (KP), asks for a subset of items whose total weight does not exceed the knapsack capacity, and whose profit is a maximum. It can be formulated as follows:

$$\max \sum_{j \in N} p_j \, x_j \tag{1.1}$$

$$\sum_{j \in N} w_j \, x_j \leq c \tag{1.2}$$

$$x_j \in \{0,1\} \quad j \in N \tag{1.3}$$

Each variable x_j takes value 1 iff item j is inserted in the knapsack. This problem is NP-hard, although solvable in pseudopolynomial time by dynamic programming. (KP) has been widely studied in the literature because of its practical and theoretical relevance, and because it arises as a subproblem in many more complex problems (see, e.g., Kellerer, Pferschy and Pisinger [15]).

For convenience we assume $c = 1$ and that items are sorted in nonincreasing order of efficiencies $\frac{p_j}{w_j}$. Moreover, for a given set S of items, we will denote by $p(S) = \sum_{j \in S} p_j$ and $w(S) = \sum_{j \in S} w_j$ the total profit and weight, respectively, of the items in S.

2. The Uncertain Context. When considering uncertain settings, our starting point is the mentioned work by Bertsimas and Sim [8]. Accordingly, we study situations in which the exact value of each coefficient $\widetilde{w}_j$ ($j \in N$) is not known in advance, but belongs to an interval $[w_j - \underline{w}_j, w_j + \overline{w}_j]$ and at most Γ coefficients can change from their nominal value w_j to an arbitrary value in the interval. Parameter Γ denotes the level of guarantee with respect to data uncertainty we want to have. Indeed, this definition of uncertainty reflects the fact that it is very unlikely that all $\widetilde{w}_j$ coefficients take simultaneously their worst value, and that whenever more than Γ coefficients change, they tend to balance each other. We assume Γ to be an integer value, since it represents the number of coefficients that change with respect to their nominal value. Since the problem becomes a nominal (KP) for $\Gamma \geq n$, we can assume w.l.o.g. that $\Gamma < n$. In the following we will always assume, in accordance with [8], that all profit coefficients are known in advance, i.e., no uncertainty is allowed on such data. Indeed, if uncertainty affects coefficients in the objective function, one can still bring the objective function in the constraint matrix and optimize an equivalent problem – although the resulting problem is no more a knapsack problem. Similarly, we also assume that the exact value of the knapsack capacity is known; otherwise a dummy item with a very large profit can be introduced to reflect uncertainty on the right-hand side in (1.2) – although this may change approximability results.

Finally, we make the natural assumption that each item can be inserted in the knapsack as a robust solution on its own, i.e.,

$$w_j + \overline{w}_j \leq 1 \quad \forall j \in N. \tag{2.1}$$

Clearly, any item violating this assumption could be immediately removed from consideration. Assuming that the actual weight of each item belongs to some interval is without loss of generality: indeed, the actual weight of each item j must be positive, whereas assumption (2.1) implicitly defines an upper bound on $\overline{w}_j$. Note that the interval associated with each item j is not required to be symmetric with respect to the nominal value w_j.

The resulting problem has been defined as *Robust Knapsack Problem* (RKP) in [8], in which the *price of robustness* concept is introduced. This refers to the ratio

between the nominal and the robust solution values, and measures the loss in the objective function incurred by the requirement to provide a robust solution. As already mentioned, the price of robustness has already been evaluated from a computational and probabilistic point of view. In the present paper we provide a missing yet very important piece of information concerning its analysis, namely we describe it from a worst-case perspective for both (RKP) and for its fractional relaxation. In addition, as it is usually done for combinatorial optimization problems, we also study the worst-case performance ratio of both the fractional problem and of a heuristic algorithm compared to the optimal solution of (RKP). All our worst-case results are tight under certain parameter configurations. Finally, we consider a relevant special situation and present fairly involved combinatorial algorithms to compute an optimal solution for fractional (RKP) in $O(n \log n)$ time.

Computational results for the price of robustness for (RKP) with different values of Γ were given by Bertsimas and Sim [8, 7]. The latter work considers 0-1 discrete optimization problems in which uncertainty affects the objective function only, showing that, in this case, the optimal robust solution can be obtained by solving a linear number of instances of the original problem, and that the approximability of the robust problem is the same as the nominal problem.

Following the approach in [8], our notion of robustness can be obtained by replacing constraint (1.2) with the following robust constraint

$$\sum_{j \in N} w_j \, x_j + \beta(x, \Gamma) \leq 1$$

where

$$\beta(x, \Gamma) = \max_{S \subseteq N,\, |S| \leq \Gamma} \sum_{j \in S} \overline{w}_j \, x_j \tag{2.2}$$

indicates the level of protection of for a given solution x with respect to uncertainty. Expressing (2.2) as a linear program, and using duality as in [8], (RKP) can be formulated through the following MILP, with $2n+1$ variables (n of which are binary) and $n+1$ constraints.

$$\max \sum_{j \in N} p_j \, x_j \tag{2.3}$$

$$\sum_{j \in N} w_j \, x_j + \sum_{j \in N} \pi_j + \Gamma \rho \leq 1 \tag{2.4}$$

$$-\overline{w}_j \, x_j + \pi_j + \rho \geq 0 \quad j \in N \tag{2.5}$$

$$x_j \in \{0, 1\} \quad j \in N \tag{2.6}$$

$$\pi_j \geq 0 \quad j \in N \tag{2.7}$$

$$\rho \geq 0 \tag{2.8}$$

The results in [8] ensure that if weights are independent and symmetrically distributed random variables, then any feasible solution for (RKP) is feasible with a given probability, whose expression depends on Γ and may be difficult to evaluate. Pinar [22] considered 0-1 problems with uncertain cost coefficients and gave a probability bound easier to compute than that proposed in [8]. Atamturk [3] presented alternative

formulations for binary problems with uncertain objective function, providing computational experiments for some relevant problems, including (RKP). Klopfenstein and Nace [17] addressed (RKP) from a polyhedral point of view, and introduced valid inequalities that are used to solve instances with up to 20 items. A recent paper by Monaci, Pferschy and Serafini [21] studies a dynamic programming scheme for (RKP) which allows the construction of an FPTAS. Moreover, different exact approaches for (RKP) are compared from a computational point of view.

Another related variant of (KP) studied in the literature is known as the *Chance-constrained Knapsack Problem* (CKP), i.e., the stochastic problem in which weights are subject to some probability distribution and the capacity constraint must be satisfied with a given probability, say $1 - \varepsilon$. Klopfenstein and Nace [16] showed that if weights are independent and identically distributed random variables and the width of the uncertain interval $[\overline{w}_j - \underline{w}_j]$ is the same for all items, then there exists a Γ for which any feasible solution for (RKP) is feasible for (CKP) as well. Fortz, Labbé, Louveaux and Poss [13] considered a variant of the problem in which a recourse variable is used to penalize capacity violation, and proved that the resulting problem is NP-hard in general; then, they proved that for some relevant special cases the problem is polynomially solvable. In the *Recoverable Knapsack Problem*, one is required to produce a solution that is not necessarily feasible under uncertainty, but whose feasibility can be recovered by means of some legal moves. Büsing, Koster and Kutschka [9, 10] addressed this problem and defined legal moves as the removal of at most K items from the solution, so as to model a telecommunication network problem. For this problem, they gave different ILP formulations, cut separation procedures and computational experiments.

The remaining contributions on uncertain knapsack in the literature consider the case in which uncertainty belongs to a given set of scenarios, yielding to a max-min knapsack (MNK) problem. Yu [25] considered the case in which uncertainty affects profit values and showed that the resulting problem is NP-hard if an infinite number of scenarios is considered, while it can be solved in pseudopolynomial time in case the number of scenarios is bounded. Lida [19] gave lower and upper bounds for (MNK) and embedded them into an exact algorithm, while Aissi, Bazgan and Vanderpooten [1] showed that (MNK) has an FPTAS in case the number of scenarios is bounded.

The paper is organized as follows. In §3 we define the worst-case price of robustness, and compute its value both for (RKP) and for the fractional (RKP). In §4 we consider the worst-case performance ratio of the fractional (RKP) and of a variant of the greedy algorithm with respect to the optimal (RKP) solution. Finally, in §5 we give an efficient algorithm to solve to optimality the fractional variant of (RKP) for a relevant special case.

3. Worst-Case Price of Robustness. In the following we will examine the worsening of the solution value due to uncertainty affecting the weight of (at most) Γ items. It is clear that such a worsening depends on some "measure" of the *uncertain set*, implicitly defined by both the value of Γ and the size of all intervals $[w_j - \underline{w}_j, w_j + \overline{w}_j]$ $(j \in N)$. We will show that, for all situations in our uncertain model, the worsening of the solution can be computed, from a worst-case perspective, as a function of two parameters only, namely Γ and $\delta_{\max}$, where

$$\delta_{\max} = \max \{\delta_j \mid j = 1, \dots, n\}$$

and $\delta_j = \overline{w}_j/w_j$ represents, for each item $j \in N$, the maximum relative weight increase with respect to the nominal weight.

Throughout the paper we will denote, for a given instance I, by $z(I)$ and $z^R(I)$ the optimal solution value of the nominal knapsack problem (1.1)–(1.3) and of its robust counterpart (2.3)–(2.8), respectively. In addition, to emphasize given values of $\delta_{\max}$ and Γ, we will extend this notation to $z^R_{\delta_{\max},\Gamma}(I)$. When no confusion arises, we will avoid explicitly indicating the instance I at hand and/or the current values for $\delta_{\max}$ and Γ.

In §3.1 we analyze the integer case, and consider, for a given instance I, the ratio $\frac{z^R_{\delta_{\max},\Gamma}(I)}{z(I)}$ that was called the price of robustness in [8]. In particular, we are interested in computing this ratio in the worst case for a fixed uncertainty, i.e.,

$$R_{\delta_{\max},\Gamma} = \inf_I \frac{z^R_{\delta_{\max},\Gamma}(I)}{z(I)}.$$

Later, in §3.2 we will study the analogous problem for the fractional (RKP), obtained when one is allowed to pack any fraction of the items, and at most Γ of them can increase their weight. The resulting problem can be modeled by dropping binary requirements (2.6) on x variables from formulation (2.3)-(2.8). For a given instance I, the resulting solution value, which is an upper bound on $z^R(I)$, will be denoted by $\overline{z}^R(I)$. Likewise, we will denote by $\overline{z}(I)$ the optimal solution value of the fractional (KP).

3.1. The Integer Case. We first observe that if assumption (2.1) is violated, one easily gets $R_{\delta_{\max},\Gamma}$ arbitrarily close to zero (for any $\delta_{\max} > 0$ and $\Gamma > 0$). Indeed consider an instance with an item having $w_1 = 1$ and profit p_1, and Γ identical items that have both profit and weight equal to $\frac{1}{\Gamma(1+\delta_{\max})}$. If we let all items have $\delta_j = \delta_{\max} > 0$, we have $z \geq p_1$ and $z^R = \frac{1}{1+\delta_{\max}}$; thus, for any $\delta_{\max}$, the ratio can be arbitrarily close to zero for sufficiently large p_1.

The following theorem provides the exact value of $R_{\delta_{\max},\Gamma=1}$ for all values of $\delta_{\max}$.

THEOREM 3.1.

$$R_{\delta_{\max},1} = \frac{1}{1+\lceil\delta_{\max}\rceil} \qquad \text{for all } \delta_{\max} > 0.$$

Proof. The upper bound on $R_{\delta_{\max},\Gamma}$ can be proved by the following instance with $n = 1 + \lceil\delta_{\max}\rceil$ items, each having $p_j = 1$, $w_j = \frac{1}{1+\lceil\delta_{\max}\rceil}$ and $\delta_j = \delta_{\max}$. It is easy to check that the nominal solution packs all items, thus $z = n$. In addition, any robust solution cannot pack more than one item, since the total weight of any two items would be $\frac{1+\delta_{\max}}{1+\lceil\delta_{\max}\rceil} + \frac{1}{1+\lceil\delta_{\max}\rceil} = \frac{2+\delta_{\max}}{1+\lceil\delta_{\max}\rceil} > 1$; thus $z^R = 1$.

We now describe a procedure that, given item set M containing the items of the optimal (KP) solution set, defines a robust solution whose profit is at least $\frac{1}{1+\lceil\delta_{\max}\rceil} z(M)$. The algorithm partitions items in M into groups, each corresponding to a feasible robust solution. Initially, each group corresponds to a single item, and we assume groups to be sorted according to nondecreasing weight. Then, the first two groups are merged, provided the resulting group is still a feasible robust solution. This process is iterated until the two lightest groups cannot be merged. Let ℓ denote the resulting number of groups, and $W_1, W_2, \ldots, W_\ell$ the associated weights.

At this point we have that $W_1 + (1 + \delta_{\max})W_2 > 1$ as in the worst case group 2 consists only of a single item which has the maximum $\overline{w}_j$ value and is inflated. Since the sum of weights over all groups (i.e., the weight of item set M) is at most 1 we have $W_1 + W_2 + ... + W_\ell \leq 1$. Because W_1 and W_2 are the two smallest groups (by weight) we get from this $W_1 + (\ell - 1)W_2 \leq 1$. Comparing these two inequalities we get $1 + \delta_{\max} > \ell - 1$. Hence, $\ell < \delta_{\max} + 2$, i.e., $\ell \leq \lceil \delta_{\max} \rceil + 1$.
Using the algorithm above we have partitioned the optimal solution into at most $1 + \lceil \delta_{\max} \rceil$ robust subsets, taking the best of them always yields a profit at least equal to $\frac{1}{1+\lceil \delta_{\max} \rceil} z(M)$. □

Note that for $\delta_{\max} \leq 1$, Theorem 3.1 yields $R_{\delta_{\max},1} = \frac{1}{2}$.
The following theorem addresses the case of general $\Gamma \geq 2$.

THEOREM 3.2.

$$R_{\delta_{\max},\Gamma} = \frac{1}{1 + \lceil 2\delta_{\max} \rceil} \qquad \text{for all } \delta_{\max} > 0,\ \Gamma \geq 2.$$

Proof. To prove the upper bound, consider the following extension of the instance given in the proof of Theorem 3.1: There are $n = 1 + \lceil 2\delta_{\max} \rceil$ identical items, each having profit one, weight equal to $\frac{1}{1+\lceil 2\delta_{\max} \rceil}$ and $\delta_j = \delta_{\max}$. It is easy to see that the nominal solution can pack all items, whereas any pair of items is just above the bound required for robustness, since their weight is $2(1 + \delta_{\max})\frac{1}{1+\lceil 2\delta_{\max} \rceil} > 1$. Hence $z = 1 + \lceil 2\delta_{\max} \rceil$ and $z^R = 1$, which yields that $R_{\delta_{\max},\Gamma}$ cannot be larger than $\frac{1}{1+\lceil 2\delta_{\max} \rceil}$.

To prove the lower bound we use the same algorithm as in the proof of Theorem 3.1, i.e., we iteratively merge the two smallest groups of items as long as their union remains a feasible robust solution. At the end of this process we have a partition of M into ℓ groups with $W_1 + W_2 + ... + W_\ell \leq 1$. Again the sorting of the groups by total weights implies

$$W_1 + (\ell - 1)W_2 \leq 1. \tag{3.1}$$

In addition, since the merging process had to be stopped we also have

$$(1 + \delta_{\max})(W_1 + W_2) > 1, \tag{3.2}$$

because from a worst-case point of view all items in W_1 and W_2 might be increased. Taking the best of these feasible groups as a robust solution yields a profit at least $1/\ell\ z(M)$, i.e., $R_{\delta_{\max},\Gamma} \geq 1/\ell$.

To show the claimed bound there must be $\ell \leq 1 + \lceil 2\delta_{\max} \rceil$. Assume by contradiction that $\ell > 1 + \lceil 2\delta_{\max} \rceil$, i.e., $\ell \geq 2 + \lceil 2\delta_{\max} \rceil$.

Recalling that groups of items are sorted by nondecreasing weight (i.e., $W_2 \geq W_1$), from (3.2) we get

$$1 < (1 + \delta_{\max})(W_1 + W_2) \leq W_1 + (1 + 2\delta_{\max})W_2$$

Since we assume $\ell - 1 \geq 1 + \lceil 2\delta_{\max} \rceil$, condition (3.1) gives

$$1 \geq W_1 + (\ell - 1)W_2 \geq W_1 + (1 + \lceil 2\delta_{\max} \rceil)W_2$$

Combining the last two inequalities one gets

$$1 \geq W_1 + (1 + \lceil 2\delta_{\max} \rceil)W_2 \geq W_1 + (1 + 2\delta_{\max})W_2 > 1$$

which is a contradiction. □

3.2. The Fractional Case. Similar to the previous section, we now consider, for a given instance I and uncertainty defined by $\delta_{\max}$ and Γ, the price of robustness of the associated fractional (RKP) and its worst-case ratio:

$$\overline{R}_{\delta_{\max},\Gamma} = \inf_I \frac{\overline{z}^R_{\delta_{\max},\Gamma}(I)}{\overline{z}(I)}.$$

We now show the following theorem:

THEOREM 3.3.

$$\overline{R}_{\delta_{\max},\Gamma} = \frac{1}{1+\delta_{\max}} \quad \text{for all } \delta_{\max} > 0,\ \Gamma \geq 1.$$

Proof. An upper bound can be shown by the following instance I. Let $n' = 1 + \lfloor \delta_{\max} \rfloor$, and define n' items with profit and weight equal to $\frac{1}{1+\delta_{\max}}$. In addition, if $n' < 1 + \delta_{\max}$, i.e., if $\delta_{\max}$ is not integer, define an additional item with profit and weight equal to $1 - \frac{n'}{1+\delta_{\max}}$. All items j have the same value $\delta_j = \delta_{\max}$ for the maximum relative weight increase. Clearly, all items can be packed in the nominal solution, i.e., $\overline{z} = 1$. Any robust solution can pack at most one item, since even for $\Gamma = 1$, no two items can be combined into a feasible solution. Hence, the price of robustness for this instance is equal to $\frac{1}{1+\delta_{\max}}$ for all $\delta_{\max} > 0$ and $\Gamma \geq 1$.

To conclude the proof, consider the optimal solution of an arbitrary instance of the fractional (KP). Mimicking Dantzig's algorithm [12] we can assume items to be sorted in nonincreasing order of efficiencies either inserting all of them or up to a total weight equal to 1 – possibly taking only a fractional part of the last item.

Consider now a fractional solution in which items are packed again according to their efficiencies, until their total weight is exactly $\frac{1}{1+\delta_{\max}}$, and let $\tilde{z}$ denote the profit of this solution. Note that $\overline{z}^R_{\delta_{\max},\Gamma} \geq \tilde{z}$ and that, by construction, this solution is robust for any $\Gamma \geq 1$. By the sorting of items, the average profit to weight ratio in this robust solution is at least as large as that of the optimal solution to fractional (KP), hence $\tilde{z} \geq \frac{1}{1+\delta_{\max}} \overline{z}$ which concludes the proof. ∎

4. Worst-Case Performance Ratio. In this section we analyze the performances of a relaxation and a heuristic algorithm for (RKP), and study the worst-case performance of the associated upper and lower bounds with respect to the optimal solution value. The following §4.1 considers the upper bound provided by the optimal solution of the fractional (RKP), whereas §4.2 introduces a lower bound determined by a variant of the well-known greedy algorithm and computes its performance ratio.

4.1. Worst-Case Performance of the Fractional (RKP) Bound. Recall that, for a given instance I, we denote by $z^R_{\delta_{\max},\Gamma}(I)$ and $\overline{z}^R_{\delta_{\max},\Gamma}(I)$ the optimal values of (RKP) and fractional (RKP), respectively.

The following theorem states that, if no restriction is imposed on $\delta_{\max}$, no approximation can be guaranteed by the upper bound provided by fractional (RKP), even in case $\Gamma = 1$. We denote the number of items n of instance I by $|I|$.

THEOREM 4.1.

$$\lim_{\delta_{\max}\to\infty} \inf_{I,|I|=n} \left\{ \frac{z^R_{\delta_{\max},\Gamma}(I)}{\overline{z}^R_{\delta_{\max},\Gamma}(I)} \right\} = \frac{1}{n} \qquad \text{for } \Gamma \in \{1,2\}.$$

$$\frac{1}{n} \leq \lim_{\delta_{\max}\to\infty} \inf_{I,|I|=n} \left\{ \frac{z^R_{\delta_{\max},\Gamma}(I)}{\overline{z}^R_{\delta_{\max},\Gamma}(I)} \right\} \leq \frac{\Gamma}{2n} \qquad \text{for all } \Gamma \geq 3.$$

Proof. First note that, by assumption (2.1), every single item can be packed as a robust solution on its own. Thus, taking the item with highest profit yields a solution with value at least $\frac{\sum_{j\in N} p_j}{n}$, hence at least $1/n$ times the optimal solution value of fractional (RKP) which proves the lower bound for any Γ and any $\delta_{\max}$.

Now we will prove an upper bound for general Γ, whose value depends on $\delta_{\max}$ and reaches $\frac{\Gamma}{2n}$ for $\delta_{\max}$ tending to infinity.

Consider the following family of instances, composed of $n \geq 2$ identical items for a fixed $\delta_{\max}$, each with profit 1 and weight $\frac{1+\varepsilon}{2\delta_{\max}+2}$, for an arbitrarily small $\varepsilon > 0$ and $\delta_j = \delta_{\max}$ for all $j \in N$.

It is clear that for $\Gamma \geq 2$ no two items can fit together in the knapsack in any integer (robust) solution, since the corresponding weight of two inflated items would be equal to $(2 + 2\delta_{\max})\frac{1+\varepsilon}{2\delta_{\max}+2} > 1$. Hence, $z^R = 1$.

As to the fractional problem, it is easy to see (e.g., by symmetry) that an optimal solution exists in which all variables take the same (fractional) value, equal to $\frac{2\delta_{\max}+2}{(n+\Gamma\delta_{\max})(1+\varepsilon)}$. Summing up all n fractional values, we immediately get the ratio

$$\frac{z^R}{\overline{z}^R} = \frac{(n + \Gamma\delta_{\max})(1 + \varepsilon)}{n(2\delta_{\max} + 2)} .$$

This ratio is arbitrarily close to

$$\frac{n + \Gamma\delta_{\max}}{n(2\delta_{\max} + 2)} \tag{4.1}$$

for sufficiently small ε.

It is easy to see that this expression is decreasing with $\delta_{\max}$. Thus, it has a maximum for $\delta_{\max} = 0$, which corresponds to the $1/2$ approximation that occurs in the nominal context. Conversely, for $\delta_{\max}$ tending to infinity the ratio tends to $\frac{\Gamma}{2n}$, which proves the statement for $\Gamma \geq 2$.

For the special case $\Gamma = 1$ we slightly modify the above instance by choosing the item weight as $\frac{1+\varepsilon}{\delta_{\max}+2}$. Since only one item must be inflated, this still guarantees that no two items fit together in a robust solution and thus $z^R = 1$.

The variables of the fractional problem now attain the value $\frac{\delta_{\max}+2}{(n+\delta_{\max})(1+\varepsilon)}$ leading to a ratio arbitrarily close to

$$\frac{n + \delta_{\max}}{n(\delta_{\max} + 2)} . \tag{4.2}$$

For $\delta_{\max}$ tending to infinity, the ratio tends to $\frac{1}{n}$ which proves the theorem. □

From the result above we can derive that the optimal solution of the fractional (KP) provides an arbitrarily bad approximation of the optimal robust solution for large enough $\delta_{\max}$.

Corollary 4.2.

$$\inf_{I,|I|=n} \left\{ \frac{z^R_{\delta_{\max},\Gamma}(I)}{\overline{z}(I)} \right\} = \frac{1}{n} \qquad \text{for } \delta_{\max} > n - 2 \text{ and all } \Gamma \geq 1.$$

Proof. The lower bound can be proved with the same argument as in the proof of Theorem 4.1. As to the upper bound, consider an instance with n identical items, having $p_j = 1$, $w_j = 1/n$ and $\delta_j = \delta_{\max} > n - 2$. We have $\overline{z} = n$ whereas no two items can be inserted in a robust solution for any $\Gamma \geq 1$, which yields $z^R = 1$. □

It is well known that the worst-case performance ratio of the fractional (KP) upper bound is equal to 1/2. The negative result of Theorem 4.1 indicates that, in the robust context, the situation is much more complicated, as the fractional problem can be arbitrarily bad in terms of approximation for large $\delta_{\max}$. Hence, a huge gap exists between the approximation guarantee that can be achieved for the nominal and the robust knapsack problems.

The following theorems indicate that a finite approximation can actually be achieved in the robust case as well, provided uncertainty on each coefficient is not "too large", i.e., $\delta_{\max} \leq 1$. This relevant case corresponds to situations in which the maximum deviation of each coefficient is not larger than the nominal value of the coefficient itself, see, e.g., the computational experiments on (RKP) in [8]. The worst-case ratio of fractional (RKP) is settled in the next Theorems 4.3 and 4.4 for the cases $\Gamma = 1$ and $\Gamma \geq 2$, respectively.

THEOREM 4.3.

$$\frac{1}{3} \leq \inf_{I,|I|=n} \left\{ \frac{z^R_{\delta_{\max},1}(I)}{\overline{z}^R_{\delta_{\max},1}(I)} \right\} \leq \frac{n + \delta_{\max}}{n(2 + \delta_{\max})} \qquad \text{for all } \delta_{\max} \leq 1.$$

Proof. The upper bound has already been proved in (4.2).

As to the lower bound, consider an optimal solution $\overline{x}$ for fractional (RKP) and let $\overline{J}$ be the set of items that are packed (possibly in a fractional way) in solution $\overline{x}$. Let y denote the solution obtained by executing Dantzig's algorithm for the nominal (KP) instance defined by item set $\overline{J}$ and capacity 1. The profit of solution y cannot be smaller than that of solution $\overline{x}$, due to item sorting and to the available capacity (1, as solution y does not take robustness into account). In addition, y has the usual structure of fractional (KP) solutions, i.e., there exists an item b such that $y_j = 1$ for $j = 1, \ldots, b-1$, $y_b \in [0, 1)$, and $y_j = 0$ for $j = b+1, \ldots, |\overline{J}|$. Let $k = \arg\max\{\overline{w}_j \,|\, j = 1, \ldots, b-1\}$ be the item with maximum weight increase that has been fully packed in solution y. Now consider an integer solution defined by item set $S = \{j \in \overline{J} \,|\, j < b, j \neq k\}$. Such a solution is robust, since the maximum weight increase due to robustness is at most equal to $\overline{w}_k \leq w_k$. Moreover, we have

$$\sum_{j \in S} p_j + p_k + p_b > \sum_{j \in \overline{J}} p_j y_j$$

Thus, taking the most profitable among solutions S, $\{k\}$ and $\{b\}$ yields a robust solution having profit at least 1/3 of the fractional (KP) solution value for any n. □

Finally, note that for n tending to infinity, the upper bound in Theorem 4.3 is equal to $\dfrac{1}{2 + \delta_{\max}}$, i.e., it is arbitrarily close to 1/3 (hence, to the lower bound) for $\delta_{\max}$ sufficiently close to 1.

THEOREM 4.4.

$$\frac{1}{4} \leq \inf_{I,|I|=n} \left\{ \frac{z^R_{\delta_{\max},\Gamma}(I)}{\overline{z}^R_{\delta_{\max},\Gamma}(I)} \right\} \leq \frac{n + \Gamma\delta_{\max}}{n(2\delta_{\max} + 2)} \quad \text{for all } \delta_{\max} \leq 1, \Gamma \geq 2.$$

Proof. The upper bound is given by (4.1).

The proof for the lower bound is similar to that of Theorem 4.3. We consider an optimal solution y to the fractional (KP) whose profit is clearly at least $\overline{z}^R_{\delta_{\max},\Gamma}$. Since $\sum_{j=1}^{b-1} w_j \leq 1$, we can proceed analogous to the proof of Theorem 3.2 and partition the item set $\{1,\ldots,b-1\}$ into at most $1+\lceil 2\delta_{\max}\rceil \leq 3$ sets, each having a weight not larger than $\frac{1}{1+\delta_{\max}}$. Each such set defines a robust solution for any value of Γ. Taking the most profitable among these solutions and item b alone, one gets a feasible solution with profit at least $\overline{z}^R_{\delta_{\max},\Gamma}/4$ for any n. □

Again, note that for large n the upper bound tends to $\dfrac{1}{2\delta_{\max}+2}$; thus, for $\delta_{\max}$ sufficiently close to 1, the bound becomes tight.

4.2. Worst-Case Performance of the Robust Greedy Algorithm. The classical greedy algorithm for (KP) works as follows: initially items are sorted according to nonincreasing efficiencies $\frac{p_j}{w_j}$. Then, they are considered one at a time, and the current item is inserted if this does not violate the capacity constraint; otherwise the item is disregarded and the next item is considered. Taking the best solution among the one produced by the algorithm and the one that packs only the most profitable item, one gets a $\frac{1}{2}$–approximation algorithm for (KP). A similar approximation for (RKP) can be derived from the results in [7] by swapping the objective function and the capacity constraint. The resulting algorithm would require $O(n)$ executions of the greedy algorithm for (KP), yielding an overall $O(n^2)$ time complexity. Actually, this number of executions can be reduced to $n-\Gamma+1$, as recently shown by [18] for (RKP) and by [2] for binary problems in which uncertainty affects a linear objective function to be optimized on a polyhedral region. However, these results do not improve the worst-case time complexity of the approach. In this section, we adapt the classical greedy algorithm for (KP) in a very natural way to take robustness into account. This intuitive heuristic approach for (RKP) has a time complexity of $O(n\log\Gamma)$, i.e., linear for fixed Γ, if the items are already sorted by efficiencies.

Our Robust Greedy algorithm is described in Fig. 4.1. For a given set S of items, S_Γ denotes the subset of (at most) Γ items with maximum *weight increase* $\overline{w}_j$. At each iteration, the current item is inserted in the knapsack and the feasibility condition is checked a posteriori, given the current set S_Γ. The addition and removal of an item from set S_Γ can be performed in $O(\log\Gamma)$ time by using a heap structure, hence the running time of the algorithm is $O(n\log\Gamma)$ plus the effort of sorting items by efficiencies.

Since we are interested in the worst-case performance of the algorithm, as in the nominal context, we also consider the solution in which only one item (the most profitable) is packed. This means that, for uncertainty defined by parameters $\delta_{\max}$ and Γ, the final heuristic solution z^H we consider has a value

$$z^H_{\delta_{\max},\Gamma} = \max\left\{z^{RG}_{\delta_{\max},\Gamma},\ \max_j\{p_j\}\right\}$$

and the worst-case performance ratio is given by

$$W^H_{\delta_{\max},\Gamma} = \inf_I\left\{\frac{z^H_{\delta_{\max},\Gamma}(I)}{z^R_{\delta_{\max},\Gamma}(I)}\right\}.$$

begin
1. Sort items according to nonincreasing efficiencies $\frac{p_j}{w_j}$;
2. $S := \emptyset$, $z^{RG} := 0$;
3. **for each** item j **do**
4. $\quad S := S \cup \{j\}$, determine S_Γ;
5. $\quad$ **if** $\sum_{j \in S} w_j + \sum_{j \in S_\Gamma} \delta_j w_j \leq 1$ **then**
6. $\quad\quad x_j := 1$, $z^{RG} := z^{RG} + p_j$;
7. $\quad$ **else**
8. $\quad\quad x_j := 0$, $S := S \setminus \{j\}$;
9. $\quad$ **endif**
10. **end for**
11. return z^{RG} and S;

end.

FIG. 4.1. *Algorithm Robust Greedy.*

It turns out that meaningful results depend on the ratio between pairs of δ_j coefficients. In particular, we introduce an additional parameter ρ as follows:

$$\rho = \frac{\delta_{\max}}{\delta_{\min}}, \qquad \delta_{\min} = \min\{\delta_j \mid j = 1, \ldots, n\}.$$

Clearly we have $\rho \geq 1$, and $\rho = 1$ iff $\delta_j = \delta_{\max}\ \forall j$. We will show that, given parameters Γ and $\delta_{\max}$, the worst-case ratio of the robust greedy algorithm can be defined as a function of ρ. Note that for arbitrary values of δ_j, i.e. unbounded ρ, the worst-case ratio can become arbitrarily small (see Theorem 4.6 for ρ tending to infinity).

In the next theorem we prove the performance bound for the case $\Gamma = 1$.

THEOREM 4.5.

$$W^H_{\delta_{\max},1}(\rho) = \min\left\{\frac{1}{1+\delta_{\max}}, \frac{1}{1+\rho}\right\} \quad \text{for all } \delta_{\max} > 0 \text{ and large enough } n.$$

Proof. As to the lower bounds, let S and S^R denote the greedy and the optimal solutions, respectively. In addition, let t be the first item (i.e., with highest efficiency) that is included in the optimal solution but is not packed by the greedy algorithm.

Since item $t \in S^R$ and each feasible solution must reserve enough capacity to accommodate the weight increase of each single item, we have

$$w(S^R) + \delta_t w_t \leq 1. \tag{4.3}$$

Let $S' = \{1, \ldots, t-1\} \cap S$ be the set of items in the greedy solution that precede item t, and k be the item with largest weight increase $\overline{w}_k$ in S'.

If $\overline{w}_k \leq \overline{w}_t$, one can easily show that $W^H_{\delta_{\max},1}(\rho) \geq \frac{1}{2}$ for arbitrary ρ: Since item t was not packed by the greedy, we have $w(S') + w_t + \delta_t w_t > 1$. Together with (4.3) this yields $w(S') + w_t > w(S^R)$.

By the sorting of efficiencies, the overall profit to weight ratio of set $S' \cup \{t\}$ is at least as high as that of S^R, which contains at most a subset of S' but also other items with lower efficiency. Thus either item set S' or item t produces a robust feasible solution having profit at least half of the optimal one.

Now we consider the case $\overline{w}_k > \overline{w}_t$. For $\Gamma = 1$ the weight of set S' fulfills $w(S') + \delta_k w_k = 1 - g$ for some $g \geq 0$ representing the unused space in the heuristic solution. Hence, we have the following lower bound for the weight of S':

$$w(S') \geq \max\{w_k, 1 - \delta_k w_k - g\}$$

As the second term is decreasing in w_k, a minimum for $W(S')$ can be obtained by setting the two expressions equal to each other. Finally, we get

$$w(S') \geq \frac{1-g}{1+\delta_k}. \tag{4.4}$$

Since the greedy could not pack item t and $\overline{w}_t < \overline{w}_k$ rules out that this was due to robustness, we conclude that $g < w_t$.

As $\overline{w}_k > \overline{w}_t$, we have $w_k > \frac{\delta_t}{\delta_k} w_t \geq \frac{\delta_{\min}}{\delta_{\max}} w_t = \frac{1}{\rho} w_t$. Thus, a second weight bound for S' is given by

$$w(S') \geq w_k > \frac{1}{\rho} w_t > \frac{1}{\rho} g\,. \tag{4.5}$$

From (4.3) we get

$$w(S^R) \leq 1 - \delta_t w_t < 1 - \delta_t\, g. \tag{4.6}$$

Note that $g < w_t$ implies $\delta_t g < \delta_t w_t < 1$ and thus this upper bound on $w(S^R)$ remains strictly greater than 0. Again, by the sorting of efficiencies, the overall profit to weight ratio of set S' is at least as high as that of S^R. Hence, we can bound the profit ratios by the weight ratios. By combining the three weight bounds (4.4), (4.5) and (4.6), we get:

$$W^H_{\delta_{\max},1}(\rho) \geq \frac{w(S')}{w(S^R)} \geq \max\left\{ \frac{\frac{1-g}{1+\delta_k}}{1-\delta_t\, g}, \frac{\frac{g}{\rho}}{1-\delta_t\, g} \right\}$$

Elementary calculus shows that, for $\delta_t \geq 1$, the first expression in the maximum is an increasing function in g for $g < 1/\delta_t$. Hence, we can plug in $g = 0$ and obtain a lower bound of $\frac{1}{1+\delta_k} \geq \frac{1}{1+\delta_{\max}}$ which proves the first part of the theorem.

For $\delta_t < 1$, the first expression is decreasing in g, while the second expression in the maximum is always increasing in g. Thus, we set the two expressions equal to each other and get a minimum for $g = \frac{\rho}{1+\delta_k+\rho}$. For this value of g the ratio is equal to

$$W^H_{\delta_{\max},1}(\rho) \geq \frac{\frac{1}{1+\delta_k+\rho}}{1-\frac{\delta_t\rho}{1+\delta_k+\rho}} = \frac{1}{1+\delta_k+\rho-\delta_t\rho} \geq \frac{1}{1+\delta_{\max}+\rho-\delta_{\min}\rho} = \frac{1}{1+\rho}\,,$$

which completes the proof of the lower bound.

As to the upper bounds, it is clear that the Robust Greedy algorithm cannot provide an approximation better than 1/2, i.e., the worst-case ratio performance of the classical greedy for (KP) which is attained for $\rho = 1$. Thus, we can assume $\rho > 1$ without loss of generality.

Consider an instance with $n+1$ items: item 1 has profit and weight equal to $1/(1+\delta_{\max})$, while all other items have profit and weight equal to $1/(n+\delta_{\max})$. Let uncertainty be defined by $\Gamma = 1$ and $\delta_j = \delta_{\max} \geq 1$ for all $j \in N$. The greedy solution packs

the first item and is done, thus $z^H = 1/(1+\delta_{\max})$. The optimal solution inserts the n identical items, with total weight equal to 1 (including robustness for $\Gamma = 1$), and overall profit $z^R = n/(n+\delta_{\max})$. For n tending to infinity, the $\frac{1}{1+\delta_{\max}}$ upper bound follows.

Finally, consider an instance with $n = \rho + 2$ items, each having a profit of 1 and weight equal to $1/(1+\alpha+\rho)$ for some $1 < \alpha \leq \frac{\rho}{\rho-1}$. Let maximum weight increases be defined as follows: $\delta_1 = \delta_{\max} = \alpha\rho$ and $\delta_j = \alpha$ for $j = 2, \ldots, n$. For $\Gamma = 1$ the optimal solution packs all items but item 1, with total weight equal to 1 (including robustness for $\Gamma = 1$), and overall profit $z^R = n - 1 = \rho + 1$. The greedy algorithm packs item 1, which requires a robust weight equal to $(1+\alpha\rho)/(1+\alpha+\rho)$; it is easy to see that this produces a feasible packing for any $\alpha \leq \frac{\rho}{\rho-1}$. In addition, as we assume $\rho > 1$, the residual capacity is not sufficient to allocate any additional item. Thus we have $z^H = 1$ and $W^H_{\delta_{\max},1}(\rho) = 1/(1+\rho)$. ∎

Finally, we give a general result that holds for all $\Gamma \geq 2$.

THEOREM 4.6.

$$\frac{1}{1+\delta_{\max}+\rho} \leq W^H_{\delta_{\max},\Gamma}(\rho) \leq \min\left\{\frac{1}{1+\delta_{\max}}, \frac{1}{1+\rho}\right\} \quad \begin{array}{l}\text{for all } \delta_{\max} > 0, \Gamma \geq 2 \\ \text{and large enough } n.\end{array}$$

Proof. As to the lower bound, let S, S^R, t, S' be defined as in the proof of Theorem 4.5. Again, since items are sorted by nonincreasing efficiencies, we will bound the ratio among the solution values by the ratio among the solution weights.

Let us denote by $\overline{S}^R \subseteq S^R$ the set of (at most) Γ items with maximum weight increase in the optimal solution S^R. By robustness the optimal solution satisfies

$$w(S^R) + \sum_{j \in \overline{S}^R} \delta_j w_j \leq 1 \tag{4.7}$$

As to the heuristic solution, let us first assume that either $|S'| < \Gamma$ or that item t is one of the Γ items with maximum weight increase in $S' \cup \{t\}$. This means that item t is not packed in the heuristic solution as it could not be inserted with its robust weight. We now denote by $\overline{S}'$ the set of (at most) $\Gamma - 1$ items with maximum weight increase in S', with $\overline{S}' = S'$ if $|S'| < \Gamma$.

Since item t is not inserted by the greedy, we have

$$1 < w(S') + \sum_{j \in \overline{S}'} \delta_j w_j + (1+\delta_t) w_t \leq w(S')(1+\delta_{\max}) + (1+\delta_t) w_t,$$

which can be written as

$$(1+\delta_{\max}) w(S') > 1 - (1+\delta_t) w_t. \tag{4.8}$$

We now distinguish two cases.

Case (i): $\sum_{j \in \overline{S}^R} \delta_j w_j \geq \delta_{\max} w(S')$.

According to (4.7) we have $w(S^R) \le 1 - \sum_{j \in \overline{S}^R} \delta_j w_j \le 1 - \delta_{\max} w(S')$ and thus, combining with (4.8), we get $w(S^R) - w(S') \le 1 - (1+\delta_{\max}) w(S') < (1+\delta_t) w_t$. Since items are sorted by efficiencies, the optimal solution value cannot exceed $P(S^R) \le P(S') + \left[w(S^R) - w(S')\right] \frac{p_t}{w_t} \le P(S') + (1+\delta_t) p_t \le (2+\delta_t) z^H \le (2+\delta_{\max}) z^H \le (1+\rho+\delta_{\max}) z^H$.

Case (ii): $\sum_{j \in \overline{S}^R} \delta_j w_j < \delta_{\max} w(S')$.

Since item t belongs to the optimal solution, we have $\delta_{\max} w(S') > \sum_{j \in \overline{S}^R} \delta_j w_j \ge \delta_t w_t$ which yields

$$w(S') > \frac{\delta_t}{\delta_{\max}} w_t. \tag{4.9}$$

On the other hand we get from (4.8)

$$w(S') > \frac{1 - (1+\delta_t) w_t}{1 + \delta_{\max}}. \tag{4.10}$$

We also have

$$w(S^R) \le 1 - \sum_{j \in \overline{S}^R} \delta_j w_j \le 1 - \delta_t w_t. \tag{4.11}$$

Combining the three weight bounds (4.9), (4.10) and (4.11) and bounding profit ratios by weight ratios we get

$$W^H_{\delta_{\max},\Gamma}(\rho) \ge \frac{w(S')}{w(S^R)} \ge \max \left\{ \frac{\frac{\delta_t}{\delta_{\max}} w_t}{1 - \delta_t w_t}, \frac{1}{1+\delta_{\max}} \frac{1 - (1+\delta_t) w_t}{1 - \delta_t w_t} \right\}.$$

Elementary calculus shows that the first expression in the maximum is always an increasing function in w_t, whereas the second expression is always decreasing in w_t. Setting the first two terms equal to each other we get that the minimum ratio of these functions is attained at $w_t = \frac{\delta_{\max}}{2\delta_{\max}\delta_t + \delta_{\max} + \delta_t}$. For this w_t value, the ratio is equal to

$$W^H_{\delta_{\max},\Gamma}(\rho) \ge \frac{\frac{\delta_t}{2\delta_{\max}\delta_t + \delta_{\max} + \delta_t}}{1 - \frac{\delta_{\max}\delta_t}{2\delta_{\max}\delta_t + \delta_{\max} + \delta_t}} = \frac{\delta_t}{\delta_{\max}\delta_t + \delta_{\max} + \delta_t}$$

Since this lower bound is increasing in δ_t, we can substitute δ_t by $\delta_{\min} = \frac{\delta_{\max}}{\rho}$ and get immediately

$$W^H_{\delta_{\max},\Gamma}(\rho) \ge \frac{\frac{\delta_{\max}}{\rho}}{\delta_{\max} + (\delta_{\max}+1)\frac{\delta_{\max}}{\rho}} = \frac{1}{1 + \delta_{\max} + \rho}.$$

In case S' contains (at least) Γ items having a weight increase not smaller than $\overline{w}_t$, one can easily prove the result in a similar way. Indeed, since item t is not packed by the greedy with its nominal weight, inequality (4.8) can be strengthened as

$$(1+\delta_{\max}) w(S') > 1 - w_t,$$

which in both cases (i) and (ii) leads to results that are even stronger than before.

We now prove the upper bound values. Consider the following instance with $n+1$ items: item 1 has profit and weight equal to $1/(1+\delta_{\max})$, while all other items have profit and weight equal to $1/(n+\Gamma\alpha)$ for some positive α. Let the maximum percentage weight increases be defined as follows: $\delta_1 = \delta_{\max}$, and $\delta_j = \alpha$ for $j = 2, \ldots, n+1$. Finally, assume $n > \Gamma$ and $\delta_{\max} \geq \alpha$, thus $\rho = \delta_{\max}/\alpha$. As the robust weight of the first item is 1, the greedy solution packs this item and is done, yielding $z^H = 1/(1+\delta_{\max})$. The optimal solution inserts the n identical items: Γ of them are inflated, with a total weight equal to $\Gamma(1+\alpha)/(n+\Gamma\alpha)$, while $n-\Gamma$ are packed with their nominal weight, with a total weight equal to $(n-\Gamma)/(n+\Gamma\alpha)$. Thus the optimal solution has a value $z^R = n/(n+\Gamma\alpha)$ and

$$W^H_{\delta_{\max},\Gamma}(\rho) = \frac{1/(1+\delta_{\max})}{n/(n+\Gamma\alpha)} = \frac{n+\Gamma\alpha}{n(1+\delta_{\max})}$$

which is arbitrarily close to $\frac{1}{1+\delta_{\max}}$, for any fixed Γ and α, for large enough n. Taking $\alpha = 1$ we get $\rho = \delta_{\max}$ and the second upper bound follows. □

5. Exact Solution of the Fractional (RKP). In this section we discuss a relevant special case of (RKP) arising when all coefficients δ_j take the same value $\delta_{\max}$. As seen in the previous sections, this is a very relevant case from a worst-case analysis perspective, in the sense that all upper bounds on the worst-case price of robustness were attained for instances having this property. Also from a practical point of view, it is counterintuitive to the notion of uncertainty to assume that while the exact value of the weight may be unclear, the interval can be specified individually for each item. It is much more natural to assume a certain percentage range describing the possible range of the weight value, much like a confidence interval. Finally, we note that this special case follows exactly the setup introduced by Bertsimas and Sim [8, Sec. 6.1] when the Robust Knapsack problem was introduced, and has been addressed in many other papers; e.g., Fortz and Poss [14] considered a stochastic knapsack problem in which the weight of each item is a random variable and gave a reformulation of the problem in case each weight follows a gaussian distribution where the variance is correlated with the mean value and in case each weight follows a Gamma distribution with the same scale parameter. Therefore, for the rest of this section, we will assume that $\delta_j = \delta_{\max}$ for all items j.

We will consider the fractional (RKP), i.e., the relaxation that arises when one is allowed to take any fraction of the items and at most Γ coefficients change up to their worst value. For this problem, obtained by removing integrality constraints from model (2.3)–(2.8), we analyze the structure of an optimal solution and present an $O(n \log n)$ algorithm for its solution. This natural relaxation of (RKP) with divisible items provides a fast way for obtaining upper bounds to be incorporated into exact solution frameworks but is also of theoretical interest in its own right.

5.1. Fractional (RKP) Solution Structure. In this section we characterize the structure of any optimal fractional solution. Based on this result, we will then present an algorithm that computes an optimal solution as a function of the smallest weight $w_{\min}$ among the Γ largest items in the solution. For the sake of simplicity, we first provide in §5.2 an algorithm that addresses a special case; finally we give in §5.3 the algorithm for the general case.

In the following, we will denote by S_Γ the set of at most Γ items with highest weight in the optimal fractional solution (meaning weight in the solution, i.e., $w_i\,\overline{x}_i$ –

which may be less than w_i). These are exactly the items which consume an increased weight due to robustness. Ties are broken by assigning to S_Γ the items with lowest indices (i.e., highest efficiencies) among those contributing the same weight. By definition there is

$$w_{\min} = \min\{w_j \, \overline{x}_j \mid \overline{x}_j > 0 \text{ and } j \in S_\Gamma\}.$$

As usual we exclude trivial cases and assume that the capacity constraint is fulfilled with equality (taking also robustness into account).

Based on the sorting of items by efficiency, we can informally describe the solution structure described in Lemma 5.1. It consists of three parts: The most efficient items are packed completely (case 1.). The following items of medium efficiency are packed but truncated (if necessary) to contribute at most $w_{\min}$ weight (case 3a.). This means that they are good enough to be included in a nominal way but not good enough to be considered also with increased weight. The remaining items are not packed at all (case 3c.). At the border between these three subsets of items there are two break items s (case 2.) and b (case 3b.) whose weights can attain intermediate values.

The following lemma gives a formal characterization of the structure of any optimal solution to fractional (RKP).

LEMMA 5.1. *Any optimal solution $\overline{x}$ of the fractional (RKP) fulfills the following properties: There exists an index $1 \le s \le n$ such that:*

1. $\overline{x}_j = 1$ *for all* $j < s$
2. $\overline{x}_s \in [0, 1)$

If $w_s \overline{x}_s \ge w_{\min}$ *then there exists an index* b *with* $s < b \le n$ *such that:*

3a. $\overline{x}_j = \min\{1, \frac{w_{\min}}{w_j}\}$ *for all* $s < j < b$
3b. $\overline{x}_b \in [0, \min\{1, \frac{w_{\min}}{w_b}\}]$
3c. $\overline{x}_j = 0$ *for all* $j > b$

If $w_s \overline{x}_s < w_{\min}$ *then*

4. $\overline{x}_j = 0$ *for all* $j > s$

Note that 4. describes the special case where no truncated items with weight at most $w_{\min}$ exist but the first subset of fully packed items consumes almost all the capacity. In this case, only one break item s exists. The general situation is represented by 3a. to 3c. and depicted schematically in Fig. 5.1 for an instance with 9 items and $\Gamma = 3$; in this case we have $S_\Gamma = \{1, 3, 5\}$ although item 6 attains weight $w_{\min}$ as well.

Proof. The proof works by exchange arguments saying that any violation of the structure stated in the Lemma would open up a possibility for improving the solution value by increasing a certain variable while decreasing another, and in this way getting closer to the stated structure. Iterating these pair-wise exchange steps finally leads to the claimed result.

Let s denote the item with lowest index such that $\overline{x}_s < 1$, as given by case 2. Then 1. trivially holds. Following the statement of the lemma we distinguish the two main cases 3. and 4.

If $w_s \overline{x}_s \ge w_{\min}$ (case 3.) then we first consider the special case that $\overline{x}_j = 0$ for all $j > s$. In this case we set $b = s + 1$ and cases 3a., 3b. and 3c. trivially hold.

Otherwise let b be the item with highest index where $\overline{x}_b > 0$, i.e., $b > s$, and 3c. is true by definition. To prove 3b. we distinguish two cases: If $b \notin S_\Gamma$ then $w_b \overline{x}_b \le w_{\min}$

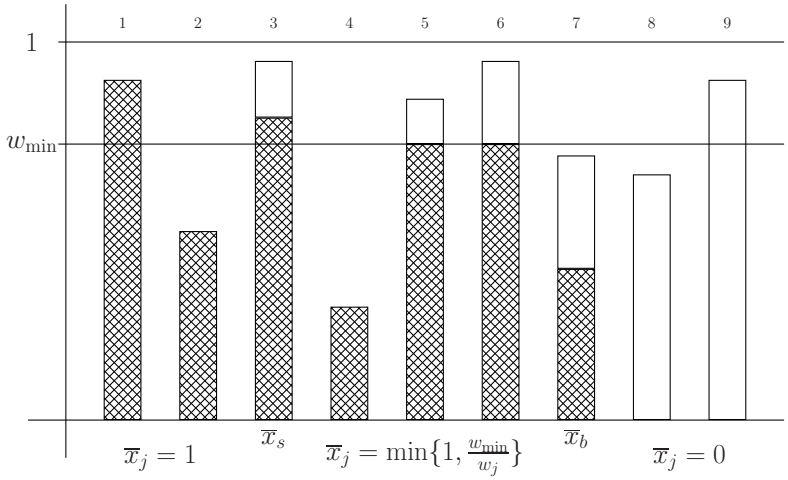

FIG. 5.1. *Solution structure of the fractional (RKP) as described by Lemma 5.1.*

by definition of $w_{\min}$, which yields $\overline{x}_b \leq \frac{w_{\min}}{w_b}$. If $b \in S_\Gamma$ then $w_b\overline{x}_b \geq w_{\min}$ by definition of $w_{\min}$. Assume that $w_b\overline{x}_b > w_{\min}$: For this case we will show that $\overline{x}_j = 1$ for all $j < b$ and thus also $\overline{x}_s = 1$ which yields a contradiction to the definition of s. Assume that $\overline{x}_j < 1$ for some $j < b$. Then we could increase $\overline{x}_j$ and decrease $\overline{x}_b$ improving the solution value (because of the sorting by efficiencies) without changing the total weight, even gaining from the robustness factor $1 + \delta_{\max}$ in case $j \notin S_\Gamma$. Thus we have shown that $w_b\overline{x}_b = w_{\min}$ for $b \in S_\Gamma$.

To conclude case 3., let us consider items j with $s < j < b$ to prove 3a. If $j \in S_\Gamma$ there must be $\overline{x}_j = \frac{w_{\min}}{w_j}$; indeed we cannot have $w_j\,\overline{x}_j < w_{\min}$ by definition of S_Γ, whereas no optimal solution can have $w_j\,\overline{x}_j > w_{\min}$, since otherwise one could increase $\overline{x}_s$ and decrease $\overline{x}_j$, producing a better solution. If $j \notin S_\Gamma$ we must have $\overline{x}_j \leq \frac{w_{\min}}{w_j}$ by definition of S_Γ. If $\overline{x}_j < \min\{1, \frac{w_{\min}}{w_j}\}$ one could increase $\overline{x}_j$ and decrease $\overline{x}_b$ to obtain a better solution, possibly gaining from the robustness factor in case $b \in S_\Gamma$. This proves 3a.

Case 4. applies for $w_s\overline{x}_s < w_{\min}$, i.e., $s \notin S_\Gamma$ by definition. Then we must have $\overline{x}_j = 0$ for all $j > s$. Otherwise we could increase $\overline{x}_s$ and decrease $\overline{x}_j$ thus increasing the total profit of the solution while keeping the total weight of the solution unchanged. This proves 4. □

5.2. Exact Solution for a Special Case. In this section we present an algorithm for solving fractional (RKP) when the optimal solution has a special structure and contains only fully packed items and at most one break item s, but no items truncated to $w_{\min}$. This corresponds to case 4. of Lemma 5.1 and covers also the case where the solution contains at most Γ items.

Of course, we do not know in advance whether this special case occurs. Hence, we have to run this algorithm in any case and compare its result (if it reaches a feasible solution) with the result computed by the general algorithm in § 5.3. The better of the two solutions will be reported as optimal solution.

If the optimal fractional solution of (RKP) contains not more than Γ items, then all of them contribute with an increased weight. Thus, if items have been sorted according to nonincreasing efficiencies, it is enough to run a greedy algorithm to compute the optimal solution for a fractional (KP) instance defined by weights $w_j(1+$

$\delta_{\max}$), and stop the computation if either the capacity bound is reached (possibly obtaining a fractional value for the last inserted item) or the first Γ items are all packed. In the former case we have obtained a candidate for the optimal solution with at most Γ items and stop the algorithm. In the latter case, the first Γ items did not consume all the capacity. Thus, to define a solution fulfilling the structure described in case 4. of Lemma 5.1, we extend this solution in the following way.

Let S_Γ denote the Γ items contributing the highest weights in the current solution (initially, $S_\Gamma = \{1, \dots, \Gamma\}$) and $w_{\min}$ be the smallest weight of an item in the current set S_Γ. S_Γ and $w_{\min}$ will be updated during the further iterations. For the current item j (starting with $j = \Gamma + 1$) we initially have $x_j = 0$ and consider two cases:

- If $w_j \leq w_{\min}$, item j cannot enter set S_Γ; thus, we set x_j as large as possible depending on the remaining capacity for the nominal weight w_j. If $x_j = 1$ we iterate the procedure with the next item $j + 1$. If $x_j < 1$ we have consumed all the available capacity and stop the iteration, as we have reached a feasible solution for case 4. of Lemma 5.1.
- If $w_j > w_{\min}$, we want to increase x_j as much as possible but have to take into account that j may enter S_Γ replacing an item, say k, having weight $w_k = w_{\min}$. Therefore, we first try to pack item j completely with its robust weight $w_j(1 + \delta_{\max})$ at the same time reducing item k to its nominal weight (although we keep $x_k = 1$). This requires an additional capacity equal to $w_j + \delta_{\max}(w_j - w_k)$.
  If the available capacity allows this option, j replaces k in S_Γ and we iterate with the next item $j + 1$.
  Otherwise, the capacity is too small and we keep item k in S_Γ, i.e. k is packed with its increased weight, and set x_j as large as possible according to its nominal weight. If this yields a solution with $w_j x_j < w_{\min}$ we have again reached a feasible solution for case 4. of Lemma 5.1 and stop the algorithm. Otherwise, i.e., $w_j x_j \geq w_{\min}$, we have determined a solution structure according to case 3. of Lemma 5.1 with $s = j$, $b = j + 1$ and $x_b = 0$. However, such a case will be considered any way in the main algorithm (see §5.3) and we discard the solution. Note that if $w_j x_j > w_{\min}$ our solution is not correct in this case, since we would have to consider the increased weight for j.

A structured description of the algorithm is depicted in Fig. 5.2. This algorithm either computes a feasible solution with the properties described in case 4. of Lemma 5.1 or determines that no such solution exists (and thus terminates without solution). In the former case, the proposed solution has to be compared to the outcome of the main algorithm, and the better of the two w.r.t. their objective function value will be reported as final optimal solution for fractional (RKP).

The above algorithm can be implemented to run in $O(n \log \Gamma)$ time after sorting. While the first phase (considering the first Γ items) is trivial, the second phase requires access to the item k having the lowest weight $w_k = w_{\min}$ in S_Γ. Therefore, we represent S_Γ by a balanced binary tree (see e.g., [11]) sorted by weights, so as to perform the possible swap among items j and k in $O(\log \Gamma)$ time.

5.3. Exact Solution for the General Case. In this section we address the general case for the optimal solution structure, corresponding to case 3. of Lemma 5.1. The algorithm is based on the following observation: If we knew the value $w_{\min}$, i.e. the smallest weight among the Γ largest items in the solution, we could compute the associated solution according to the structure given in Lemma 5.1 (if any). Since we

```
begin
1.  Solve the fractional (KP) instance defined by the first Γ items with
    increased weights and let c̄ denote the residual capacity;
2.  if c̄ = 0 then return the current solution;
3.  define current item set S_Γ and weight w_min;
4.  for j = Γ + 1, ..., n do
5.      if w_j ≤ w_min then
6.          pack item j with its nominal weight as much as possible;
            update c̄ accordingly;
7.          if c̄ = 0 then return the current solution;
8.      else
9.          if c̄ ≥ w_j + δ_max(w_j − w_min) then
10.             pack item j with increased weight and update S_Γ,
                w_min and c̄ accordingly;
11.             if c̄ = 0 then return the current solution;
12.         else
13.             pack item j with its nominal weight as much as possible;
14.             if w_j x_j ≥ w_min then discard the solution and return;
15.         endif
16.     endif
17. end for
18. return the current solution;
end.
```

FIG. 5.2. *Algorithm for fractional (RKP) solutions with a special structure.*

do not know $w_{\min}$, the main task of the algorithm is a search for $w_{\min}$. Therefore, we consider throughout the algorithm a candidate value w' for $w_{\min}$ with an associated feasible solution (if it exists) and let w' pass through the whole range of possible weight values, i.e. any real number between 0 and the largest item weight. We will do this in decreasing order of weights.

Associating with each candidate value w' the objective function value of the corresponding solution we obtain a piecewise-linear profit function $P(w')$. Taking the maximum of $P(w')$ yields a candidate for the optimal solution of (RKP). Note that this result still has to be compared to the best solution derived for the special case in § 5.2 and the better of the two constitutes the actual optimal solution of the fractional (RKP).

Section 5.3.1 shows how to determine the initial (i.e., largest) value for $w_{\min}$. Then, the search continues by considering in one step all candidate values w' in one search interval $[w^1, w^2]$ defined by two neighboring item weights $w^1 < w^2$, i.e. no item has a weight between w^1 and w^2. This will be described in § 5.3.2. Then we show in § 5.3.3 how to concatenate these search intervals in decreasing order, i.e. how to move from one search interval $[w^1, w^2]$ to the adjacent interval $[w^0, w^1]$ in an efficient way. To give a better overview of the general structure of our algorithm we summarize the main steps in Fig. 5.3. A detailed running time analysis will be given in § 5.3.4.

5.3.1. Finding a Feasible Solution. In this section we describe how to check whether a given item weight w_j is a feasible candidate w' for $w_{\min}$, or whether no solution fulfilling the required structure with $w_{\min} = w_j$ exists. In the latter case a

begin
define a candidate solution having a special structure
1. let U_0 be the solution value produced by the algorithm of Fig. 5.2;

define the initial solution having the general structure
2. use binary search on the item weights to determine the maximum w' value permitting a fractional solution as described in Lemma 5.1;
3. compute the optimal fractional solution with value U_1 for w';
4. initialize the set structure and set the starting interval with $w^2 = w'$;
5. **for all** intervals $[w^1, w^2]$ of two consecutive item weights **do**

update the current solution
6. compute the residual capacity c_r when w' decreases from w^2 to w^1;
7. iteratively increase the variable associated with the most profitable item taking robustness into account until c_r is used up, keep the set structure updated and possibly update the incumbent U_1;

define the initial solution for the next interval
8. update the set structure for all items with weight w^1
9. **end for**
10. return the solution with maximum value among U_0 and U_1.

end.

FIG. 5.3. *Outline of the algorithm for solving fractional (RKP).*

smaller candidate for $w_{\min}$ has to be chosen, while in the former case also a larger candidate may exist.

This procedure will be incorporated into a binary search scheme which searches for the largest possible item weight w^t which can be used as candidate value w'. This will define the first search interval to be considered by the algorithm given in § 5.3.2.

Given an item weight w_j, we go through the items in increasing order of efficiencies, i.e., indices, and set the following temporary solution for each item k: If $w_k < w_j$ set $x_k = 1$, otherwise truncate item k, i.e., set $x_k = \frac{w_j}{w_k}$, and put k into the temporary set S_Γ.

We stop this procedure as soon as $|S_\Gamma| = \Gamma$, i.e., Γ items with weight at least w_j are found, and set all remaining variables equal to 0. If we reach the end of the item list without detecting Γ items of weight at least w_j, we conclude that the candidate value w_j is too large and does not permit a feasible solution according to Lemma 5.1.

Otherwise we compute the total weight of the temporary solution and check whether it fulfills the weight constraint, i.e.,

$$\sum_{i \in S_\Gamma} w_i(1 + \delta_{\max})\, x_i + \sum_{i \notin S_\Gamma} w_i x_i \leq 1. \tag{5.1}$$

If this condition is violated it follows from Lemma 5.1 that the optimal solution can not contain Γ items with weight at least w_j, since our construction so far determined a solution with minimal weight for the required structure. Again, the candidate value w_j is too large for permitting a feasible solution according to Lemma 5.1.

If condition (5.1) holds, w_j is a feasible candidate for $w_{\min}$ and we continue binary search to check whether also candidate values larger than w_j exist. Finally, consider

the w_j value corresponding to the largest feasible candidate for $w_{\min}$, and let c_r denote the residual knapsack capacity for the associated temporary solution. We first show how to extend the temporary solution to consume this remaining capacity.

Two variables are immediate candidates for an increase: On one hand, we might increase the value of a truncated variable. Clearly, the most efficient of them is the best choice. Hence, we define item s (in accordance with Lemma 5.1) as the smallest index of a truncated variable, i.e. $s := \min\{i \mid x_i < 1, i \in S_\Gamma\}$. The special case that no such s exists will be discussed further below.

On the other hand, we might also increase a variable currently set to 0. Again we would choose the smallest index b of such a variable defined as $b := \min\{i \mid x_i = 0\}$.

It remains to decide whether x_s or x_b should be increased. Taking into account that $s \in S_\Gamma$ and thus item s consumes an increased weight, this decision depends on the following comparison:

$$\max\left\{\frac{p_s}{w_s(1+\delta_{\max})}, \frac{p_b}{w_b}\right\}. \tag{5.2}$$

Depending on the outcome of this comparison we distinguish two cases and their subcases.

Case (i), s yields the maximum in (5.2)[1]:
The increase of x_s has two possible limitations:

Case (ia), $(w_s - w_j)(1+\delta_{\max}) \geq c_r$:
In this case x_s can absorb all the available capacity c_r. Taking into account robustness we set $x_s = \frac{w_j}{w_s} + \frac{c_r}{w_s(1+\delta_{\max})} \leq 1$ and stop.

Case (ib), $(w_s - w_j)(1+\delta_{\max}) < c_r$:
Now x_s can be set to 1 leaving a reduced residual capacity $c_r > 0$. Thus we iterate the process and compare $s+1$ to b.

Case (ii), b yields the maximum in (5.2):
We set $x_b = \min\{\frac{c_r}{w_b}, \frac{w_j}{w_b}, 1\}$ and consider the three resulting cases for x_b:

Case (iia), $x_b = \frac{c_r}{w_b}$:
In this case x_b can absorb all the available capacity c_r and we stop.

Case (iib), $x_b = \frac{w_j}{w_b}$:
Variable x_b is truncated and we continue to consume the reduced residual capacity c_r by comparing s and $b+1$.

Case (iic), $x_b = 1$:
Again we continue to consume the reduced residual capacity c_r by comparing s and $b+1$.

If all items in S_Γ have variable values equal to 1 then there is no such index s. In this case we would still consume any remaining capacity by increasing the value of x_b (and possibly successive items) as far as possible as in Case (ii) above but without any comparison.

There is another more subtle special case to be considered: If there is more than one item in S_Γ with weight equal to w_j (and thus with its variable set to 1) then we could increase x_b up to 1 even if $w_b > w_j$. In such a case we would remove from S_Γ the item with lowest efficiency among those with weight equal to w_j (keeping its variable value at 1) and enter item b into S_Γ instead. Of course, we now have to

[1] This case applies also for ties.

take robustness into account for setting x_b, while the item removed from S_Γ no longer contributes an increased weight.

Note that this procedure could end with a candidate value w' which is strictly larger than all the n item weights.

This can occur if all items $i \in S_\Gamma$ have $w_i > w_j$ for the considered item weight w_j. In such a situation it may happen that all variables in S_Γ were increased to 1 by Case (ib) while using up the residual capacity, possibly except the last item in S_Γ. In such a case, we determine an increased candidate value $w' = \min\{w_i x_i \mid i \in S_\Gamma\} \geq w_j$. Hence, the first search interval in the main procedure can have an upper bound which is not equal to any of the item weights, but this does not cause any changes.

5.3.2. Search Within One Interval. In this section we consider all candidate values w' for $w_{\min}$ within one interval $[w^1, w^2]$ defined by two consecutive item weights moving in decreasing direction. In the beginning we are given a feasible solution for the candidate value $w' = w^2$ fulfilling the properties of Lemma 5.1. For the first interval, such a solution is computed by the procedure described in § 5.3.1. In the subsequent intervals, we obtain this starting solution directly from the previous iteration, as described in § 5.3.3.

As before, let S_Γ denote the set of the Γ largest items in the current solution and γ the item with highest index in S_Γ (i.e., the one with lowest efficiency), respectively. According to case 3. of Lemma 5.1 we define an item s as

$$s = \min\{j \in S_\Gamma \,|\, (x_j < 1 \text{ or } w_j = w') \text{ and } w_\ell\, x_\ell = w' \;\forall\, \ell \in S_\Gamma,\, \ell > j\}\,. \tag{5.3}$$

In general, we will have $x_s < 1$. If $x_s = 1$ and $w_s = w' = w^2$ we have the special case that the second part of S_Γ, containing items truncated to w', starts with an items s with weight exactly w'. For algorithmic clarity it makes sense to define an item $s \in S_\Gamma$ with $x_s = 1$ for this case, in a slight deviation from Lemma 5.1.

We further partition S_Γ into $S_1 = \{j \in S_\Gamma \,|\, j < s\}$, i.e. $x_j = 1$ for all $j \in S_1$, and $S_w = S_\Gamma \setminus S_1$, i.e. $w_j x_j = w'$ for all $j \in S_w$ with the exception of item s, for which $w_s x_s \geq w'$ holds. The remaining items are denoted by $E_1 = \{j \notin S_\Gamma \,|\, j < \gamma\}$ (i.e., $x_j = 1$ and $w_j < w'$ for $j \in E_1$ by case 1. of Lemma 5.1), $J_1 = \{j \notin S_\Gamma \,|\, j > \gamma, x_j = 1, w_j < w'\}$ and $J_w = \{j \notin S_\Gamma \,|\, w_j x_j = w'\}$ (case 3a.). Moreover, we possibly have an item with index $b \notin S_\Gamma$ with $x_b \in (0,1)$ and $w_b x_b < w'$. All other items have variable values 0.

A decrease of w' from w^2 to w^1 requires a decrease of all variables in $S_w \cup J_w$ (except x_s) and thus sets free some capacity of the knapsack. All variables in S_1, E_1 and J_1 remain set to 1. For simplicity of notation we will also decrease x_s although we can be sure that x_s will increase again if it contributed more than w^2 in the initial solution of this interval.

Based on the current setting of x, the maximum amount of residual capacity gained from lowering to w^1 the contribution of all variables in $S_w \cup J_w$ is given by

$$c_r = ((|S_w| - 1)(1 + \delta_{\max}) + |J_w|)(w^2 - w^1) + (w_s x_s - w^1)(1 + \delta_{\max}).$$

An illustration is given in Fig. 5.4 for the same instance depicted in Fig. 5.1. Reducing the current w' value forces items in $S_w = \{3, 5\}$ and in $J_w = \{6\}$ to reduce their weight contribution in the solution, thus freeing some capacity.

According to the solution structure described in Lemma 5.1 there are only two possibilities to utilize this gained capacity c_r: Increase either x_s or x_b. Similar to

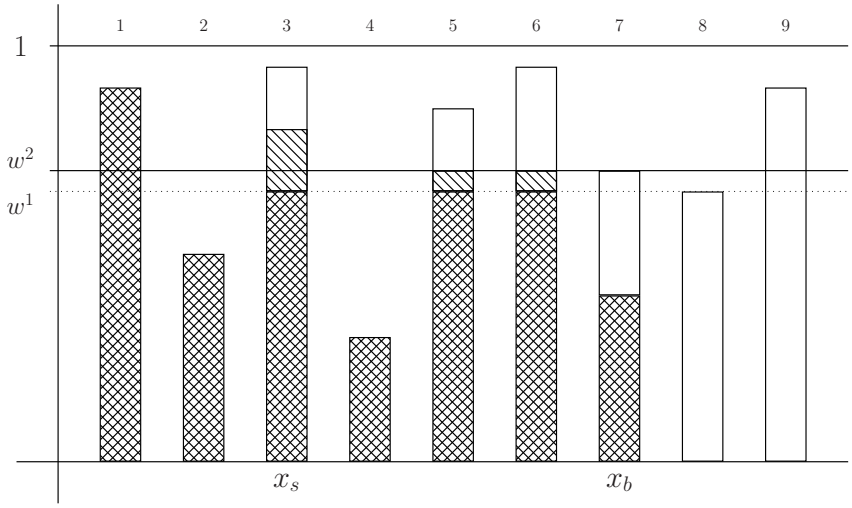

FIG. 5.4. *Decreasing w': Diagonal lines indicated the area of residual capacity c_r gained be decreasing w' from w^2 to w^1 (i.e. from w_7 to w_8 in our example).*

(5.2) we choose the alternative yielding

$$\max\left\{\frac{p_s}{w_s(1+\delta_{\max})},\ \frac{p_b}{w_b}\right\}. \tag{5.4}$$

Depending on the outcome of this comparison we distinguish two cases and their subcases. Each subcase identifies a linear piece of the objective function $P(w')$ either reaching $w' = w^1$ and thus completing the computation for the current interval, or modifying the solution structure and identifying a new item s or b for the comparison in (5.4).

Case (i), s yields the maximum in (5.4)[2], increase x_s:
The increase of x_s has two possible limitations:
Case (ia), $w^1(1+\delta_{\max}) + c_r < w_s(1+\delta_{\max})$:
In this case x_s can absorb all the available capacity c_r. After decreasing x_s together with the all other variables in S_w we have $x_s = w^1/w_s$. Now we can add the all the residual capacity and get $x_s = \frac{w^1}{w_s} + \frac{c_r}{w_s(1+\delta_{\max})}$ taking into account robustness. The condition for entering Case (ia) guarantees that $x_s < 1$. We have reached $w' = w^1$ and completed the computation for the interval $[w^1, w^2]$.
Case (ib), $w^1(1+\delta_{\max}) + c_r \geq w_s(1+\delta_{\max})$:
Now x_s reaches 1 before all the capacity c_r is consumed. We set $x_s = 1$ and move s from S_w into S_1. Then we determine the next index $s \in S_w$ and compute the new value of c_r to continue the process by evaluating (5.4) for the new choice of s.

Case (ii), b yields the maximum in (5.4), increase x_b:
Consider three possible limitations of x_b:
Case (iia), $w_b x_b + c_r < \min\{w_b, w^1\}$:
In this case x_b can absorb all the available capacity c_r without reaching w^1. We add $\frac{c_r}{w_b}$ to x_b and have reached $w' = w^1$ terminating the iteration.

[2]This case applies also for ties.

Case (iib), $w_b x_b + c_r \geq \min\{w_b, w^1\}$ and $w_b \leq w^1$:
We set $x_b = 1$ and enter b into J_1. Then we choose the next index b (which currently must have $x_b = 0$) and compute the new value of c_r to continue the process by evaluating (5.4) for the new choice of b.

Case (iic), $w_b x_b + c_r \geq \min\{w_b, w^1\}$ and $w_b > w^1$:
By definition, there is no item weight between w^1 and w^2 and hence $w_b \geq w^2$. Note that in Case (iic) there would be enough residual capacity to reach $w_b x_b > w^1$ which would violate the solution structure implied by $w' = w^1$. Instead, we decrease w' and increase $w_b x_b$ until the two values meet somewhere between w^1 and w^2. Denoting by $\hat{x}_b$ the increased value of x_b, we set equal the weight consumed by increasing x_b to $\hat{x}_b$ and the weight gained by decreasing w' from w^2 to $w_b \hat{x}_b$

$$w_b(\hat{x}_b - x_b) = ((|S_w| - 1)(1 + \delta_{\max}) + |J_w|)(w^2 - w_b\hat{x}_b) + (w_s x_s - w_b\hat{x}_b)(1 + \delta_{\max}).$$

Solving this equation for $\hat{x}_b$ we can easily obtain the new value of x_b. Now we insert b into J_w and determine the next index b to continue the process with an updated value of c_r. Note that this case applies also in the very last interval with $w^1 = 0$ (if it is reached and no earlier termination occurs by Case (2a) from § 5.3.3).

Each of the above subcases defines a linear pieces of $P(w')$ and allows the computation of a breakpoint of $P(w')$ where the maximum might be found. We determine the change of the objective function value in an efficient way. For each item j in S_1, E_1 and J_1 the contribution is simply p_j, which can be easily stored and updated for each of the three sets. For j in $J_w \cup S_w \setminus \{s\}$ we have $x_j = w'/w_j$. Then the profit contributed by j is $p_j x_j = p_j w'/w_j = (p_j/w_j)w'$. Hence we can store for each set J_w and $S_w \setminus \{s\}$ the total efficiency $\sum_j p_j/w_j$ and get as profit in the objective function $w' \cdot \sum_j p_j/w_j$. When w' is decreased, we only have to multiply the new value of w' with the factor given by the sum of efficiencies.

5.3.3. Moving From One Interval to the Next. Having completed the computation for some interval $[w^1, w^2]$ as described in § 5.3.2 we continue with the next interval $[w^0, w^1]$ for the next smaller item weight w^0. In this section we will describe the technical details of how to update the representation of the solution structure at hand at the end of the computation for $[w^1, w^2]$ to reach a valid initial solution structure for w^1. This means that the partitioning of the items into subsets defined at the beginning of § 5.3.2 has to be updated.

Our procedure depends mainly on the items with weight w^1. Let $K = \{j \mid w_j = w^1\}$. Usually, we will have $|K| = 1$, but we address the general case. Recall that γ denotes the item with highest index in S_Γ. Note that $\gamma \notin K$ since $w_\gamma > w^1$.

For all $k \in K$ in an arbitrary order we consider the following cases:

Case (1), $k > \gamma$:
This means that item k has a lower efficiency than the items in S_Γ which remains unchanged. Note that k can not be in J_w from the previous iteration: In the beginning J_w contained only items with weight at least w^2 and in Case (iic) items with weight larger than w^1 were possibly added. If $k \in J_1$ then we move k into J_w, since x_k will be decreased when w' decreases. If $k = b$ or if $x_k = 0$, all item sets remain unchanged.

Case (2), $k < \gamma$:
In this case we may have to update S_Γ which can became more complicated since a new candidate for the largest Γ items has appeared.

From the structure of the solution we deduce that $k \in E_1$. Since k is now among the Γ most efficient items with weight at least w^1, we move item k from E_1 into S_Γ and in exchange remove the current item γ from S_Γ (and from S_w) putting it into J_w. This is consistent with the definition of J_w since $w_\gamma x_\gamma = w^1$ always holds.

In the updated set S_Γ we determine the new item $\gamma \in S_\Gamma$ as the item with lowest efficiency and only refer to this new item γ from now on. However, s is not automatically updated according to (5.3) but keeps its value from the previous iteration for the time being. It will be updated explicitly further below.

We start with a subtle special case which causes an immediate termination of the algorithm. In fact, this case will usually be reached at some point when the weights of the considered interval become small.

Case (2a): $w_\gamma x_\gamma > w^1$: It follows from Lemma 5.1 for this case that all items ℓ with index smaller than γ must have $x_\ell = 1$. The fact that $w_\gamma x_\gamma > w^1$ indicates that item γ has a better efficiency than the current item b, even taking into account robustness. Otherwise, we would have kept the weight contribution of item γ equal to w' in the previous iteration. Any further decrease of w' requires a shifting of weight from items in $1, \ldots, \gamma$ to items in $b, \ldots, n$. But this can only worsen the overall solution value. Therefore, we can stop the procedure at this point and do not consider any further intervals.

Case (2b), $w_\gamma x_\gamma = w^1$:

Now we distinguish four subcases depending on item s to decide whether k belongs to S_1 or S_w, i.e., whether x_k remains fixed at 1 or will be decreased as w' goes down.

Case (2b1), $k > s$: Put k into S_w. In this case we may also have $k = \gamma$.

Case (2b2), $k < s$ and there exists $j \in S_\Gamma$ with $j > k$ and $w_j x_j > w^1$: Put k into S_1.

Case (2b3), $k < s$ and for all $j \in S_\Gamma$ with $j > k$ (including s) there is $w_j x_j = w^1$: Put k into S_w and let k be the new item s.

Case (2b4), s is not in S_Γ anymore but was removed as previous item γ:
Note that all items in the previous set S_Γ are more efficient than s and therefore contribute more than w^1 by definition of s. On the other hand, s was removed as the least efficient item in S_Γ, so no item j remains from the previous set S_Γ fulfilling the condition of Case (2b) that $w_j x_j = w^1$. There only remains the possibility that the item k newly added to S_Γ is also the new γ. Put k into S_w and let k be the new item s.

After going through these subcases for all items $k \in K$ we can proceed to evaluate the next interval $[w^0, w^1]$ as described in § 5.3.2.

5.3.4. Running Time. We now discuss the running time of the main algorithm sketched above. Sorting the items by efficiency and by weight in a preprocessing step requires $O(n \log n)$ time.

Finding a starting solution in § 5.3.1 requires a binary search with $O(\log n)$ iterations. At each iteration it suffices to go through the list of items in increasing order of efficiency, set the variable values and count the elements of S_Γ. This can be done trivially in $O(n)$ time. For the final value w' of the starting procedure we also have to assign the remaining capacity which can be done again in linear time, by keeping pointers to the current items s and b, both of which are monotonically increasing.

Then, there are $O(n)$ weight intervals $[w^1, w^2]$ to be considered in § 5.3.2. There may exist intervals requiring $O(n)$ time (e.g., assigning the available capacity c_r to

a large number of small items in Case (iic)) yielding a trivial running time bound of $O(n^2)$, but we will show that also the total running time can be bounded by $O(n \log n)$.

To this aim we store, for each item j, two labels: the first one reports the partition j belongs to, while the second one indicates whether j is in set S_Γ or not. Note that variables of items in S_w and J_w do not have to be set explicitly whenever w' changes, but their values can be easily deduced from the membership flags and the fact that all items in S_w and J_w contribute a weight of w'.

Moreover, we use pointers indicating the current items s and b for direct access. Furthermore, we keep counters updated for the cardinalities $|S_w|$ and $|J_w|$. These simply features allow the computation of the residual capacity c_r in constant time.

Considering the different subcases for an interval in § 5.3.2, it is straightforward to compute them all in constant time with the possible exception of Case (ib), where we have to find the new index s in S_w. To do this efficiently we keep the elements of S_w in a heap in decreasing order of efficiency, i.e., s is the top element of the heap. Storing the heap e.g. as a balanced binary tree, we can remove s from S_w and find the new item $s \in S_w$ in $O(\log \Gamma)$ time. Note that the at most Γ items of S_w can be inserted in such a tree in $O(\Gamma \log \Gamma)$ time, see again [11].

Moving from one interval to the next as described in § 5.3.3 boils down to elementary constant time operations. We only have to keep the data structure for S_w updated. In the beginning of Case (2) we can remove item γ from S_w in $O(\log \Gamma)$ time. In Cases (2b1), (2b3) and (2b4) item k is inserted in S_w in $O(\log \Gamma)$ time. Although there could be $|K|$ items affected in one execution of the moving procedure, each item can be inserted (and removed) at most once. Hence, the total effort of moving between intervals is bounded by $O(n \log \Gamma)$.

So far we have argued that each subcase of § 5.3.2 can be computed in $O(\log \Gamma)$ time. However, we may go through several iterations when we decrease w' from w^2 to w^1. Hence, to show the $O(n \log \Gamma)$ time bound, the total number of executed subcases must be linearly bounded. This can be shown by tracking the insertions and removals of items from sets.

Since cases (ia) and (iia) terminate the current iteration they can be executed at most n times. In Case (ib) we move an item s from S_w to S_1, while Case (iic) enters an item into J_w. It can be checked easily by going through the subcases of § 5.3.2 that items are never removed from S_1 or J_w. Hence, Case (ib) and Case (iic) can occur at most n times. In Case (iib) an item enters into J_1, but items in J_1 either stay there or are moved to J_w (Case (1)), where the stay permanently. Thus, also Case (iib) can be executed at most n times.

We conclude this section with the following theorem:

THEOREM 5.2. *The optimal fractional (RKP) solution can be computed in* $O(n \log n)$ *time for any* Γ*, if* $\delta_j = \delta_{\max}$ *for all* $j = 1, \ldots, n$.

6. Conclusions. In this paper we considered the robust knapsack problems, a variant of the knapsack problem (KP), where at most Γ items may change their weight from the given nominal value to a given upper bound. This uncertainty of item weights follows the approach of Bertsimas and Sim [8]. We considered the associated price of robustness, which was investigated in previous works only from a computational and from a probabilistic viewpoint. This work is, according to our knowledge, the first attempt to compute the price of robustness according to a worst-case analysis. We

managed to settle this question by giving exact values both for the optimal integer and fractional solutions. It turns out (not surprisingly) that the worst-case ratios depend on the maximum increase of the weight coefficients. More astonishing, the number of changed weights Γ does not have any influence on the ratios (except for the increase from $\Gamma = 1$ to $\Gamma \geq 2$). Taking the worst-case analysis further we also studied the performance ratio of natural upper and lower bounds for the optimal robust solution value and described the resulting worst-case ratios by more or less tight bounds. Our results indicate that adapting classical bounds of (KP) to the robust case does not work well in the worst-case. This might motivate the construction of new approximation algorithms and bounding procedures for (RKP).

Noting that (KP) is the simplest pure binary problem one can think of, the results in this paper provide lower bounds on the worst-case price of robustness for more complex robust optimization problems that may be studied in the future.

Finally, we mention that this kind of analysis could be performed on robust problems obtained by using different definition of robustness; e.g., the aforementioned recoverable robustness (see, Liebchen, Lübbecke, Möhring and Stiller [20]) or the recently proposed definition by Poss [23], in which the number of weights that may change is not fixed to Γ but depends on the current solution itself.

Acknowledgments. We would to thank the anonymous referees for helpful remarks and suggestions to improve the presentation of the paper.

The research was partially funded by the Austrian Science Fund (FWF): P23829

REFERENCES

[1] H. Aissi, C. Bazgan, and D. Vanderpooten, *Approximation of min-max and minmax regret versions of some combinatorial optimization problems*, European Journal of Operational Research, 179 (2007), pp. 281–290.

[2] E. Álvarez-Miranda, I. Ljubic, and P. Toth, *A note on the Bertsimas & Sim algorithm for robust combinatorial optimization problems*, 4OR, (to appear) (2013).

[3] A. Atamtürk, *Strong formulations of robust mixed 0-1 programming*, Mathematical Programming, 108 (2006), pp. 235–250.

[4] A. Ben-Tal and A. Nemirovski, *Robust convex optimization*, Mathematics of Operations Research, 23 (1998), pp. 769–805.

[5] ———, *Robust solutions of linear programming problems contaminated with uncertain data*, Mathematical Programming, 88 (2000), pp. 411–424.

[6] D. Bertsimas, D. Brown, and C. Caramanis, *Theory and applications of robust optimization*, SIAM Review, 53 (2011), pp. 464–501.

[7] D. Bertsimas and M. Sim, *Robust discrete optimization and network flows*, Mathematical Programming, 98 (2003), pp. 49–71.

[8] ———, *The price of robustness*, Operations Research, 52 (2004), pp. 35–53.

[9] C. Büsing, A.M.C.A. Koster, and M. Kutschka, *Recoverable robust knapsacks: Γ-scenarios*, in Proceedings of INOC 2011, International Network Optimization Conference, J. Pahl, T. Reiners, and S. Voß, eds., Lecture Notes in Computer Science 6701, Springer-Verlag, Berlin Heidelberg, 2011, pp. 583–588.

[10] ———, *Recoverable robust knapsacks: the discrete scenario case*, Optimization Letters, 5 (2011), pp. 379–392.

[11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms, 2nd ed.*, MIT Press, 2001.

[12] G.B. Dantzig, *Discrete variables extremum problems*, Operations Research, 5 (1957), pp. 266–277.

[13] B. Fortz, M. Labbé, F. Louveaux, and M. Poss, *Stochastic binary problems with simple penalties for capacity constraints violations*, Mathematical Programming, (to appear).

[14] B. Fortz and M. Poss, *Easy distributions for combinatorial optimization problems with probabilistic constraints*, Operations Research Letters, 38 (2010), pp. 545–549.

[15] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*, Springer, Berlin, Germany, 2004.

[16] O. Klopfenstein and D. Nace, *A robust approach to the chance-constrained knapsack problem*, Operations Research Letters, 36 (2008), pp. 628–632.

[17] ———, *Cover inequalities for robust knapsack sets - application to the robust bandwidth packing problem*, Networks, 59 (2012), pp. 59–72.

[18] C. Lee, K. Lee, K. Park, and S. Park, *Technical note – branch-and-price-and-cut approach to the robust network design problem without flow bifurcations*, Operations Research, 60 (2012), pp. 604–610.

[19] H. Lida, *A note on the max-min 0-1 knapsack problem*, Journal of Combinatorial Optimization, 3 (1999), pp. 89–94.

[20] C. Liebchen, M. Lübbecke, R.H. Möhring, and S. Stiller, *The concept of recoverable robustness, linear programming recovery, and railways applications*, in Robust and Online Large-Scale Optimization, R.K. Ahuja, R. Moehring, and C. Zaroliagis, eds., Lecture Notes in Computer Science 5868, Springer-Verlag, Berlin Heidelberg, 2009, pp. 1–27.

[21] M. Monaci, U. Pferschy, and P. Serafini, *Exact solution of the robust knapsack problem*. submitted manuscript, available as Optimization Online 2012-12-3703, 2012.

[22] M.C. Pinar, *A note on robust 0-1 optimization with uncertain cost coefficients*, 4OR, 2 (2004), pp. 309–316.

[23] M. Poss, *Robust combinatorial optimization with variable budgeted uncertainty*, 4OR, 11 (2013), pp. 75–92.

[24] A.L. Soyster, *Convex programming with set-inclusive constraints and applications to inexact linear programming*, Operations Research, 21 (1973), pp. 1154–1157.

[25] G. Yu, *On the max-min 0-1 knapsack problem with robust optimization applications*, Operations Research, 44 (1996), pp. 407–415.