Optimization over Integers with Robustness in Cost and Few Constraints

Kai-Simon Goetzmann[1]*, Sebastian Stiller[2]**, and Claudio Telha[3]

[1] goetzman@math.tu-berlin.de, Institut für Mathematik, Technische Universität Berlin
[2] sebastia@mit.edu, Sloan School of Management, Massachusetts Institute of Technology
[3] ctelha@mit.edu, Operations Research Center, Massachusetts Institute of Technology

Abstract. Robust optimization is an approach for optimization under uncertainty that has recently attracted attention both from theory and practitioners. While there is an elaborate and powerful machinery for continuous robust optimization problems, results on robust combinatorial optimization and robust linear integer programs are still rare and hardly general. In a seminal paper Bertsimas and Sim [7] show that for an arbitrary, linear 0-1-problem, over which one can optimize, one can also optimize the cost-robust counterpart. They explicitly note that this method is confined to binary problems. We present a result of this type for general integer programs. Further, we extend the result to integer programs with uncertainty in one constraint.

We show that one can optimize a not necessarily binary, *cost* robust problem, for which one can optimize a slightly modified version of the deterministic problem. Further, in case there is a ρ–approximation for the modified deterministic problem, we give a method for the cost robust counterpart to attain a $(\rho + \varepsilon)$–approximation (for minimization problems; for maximization we get a 2ρ–approximation), or again a ρ–approximation in a slightly more restricted case.

We further show that general integer linear programs where a single or few *constraints* are subject to uncertainty can be solved, in case the problem can be solved for constraints on piecewise linear functions. In case the programs are again binary, it suffices to solve the underlying non-robust program $n + 1$ times.

To demonstrate the progress achieved by our general techniques for non-binary integer programs, we apply them to two classes of integer programs, namely, totally unimodular integer programs and integer programs with two variables per inequality. We also exemplify the power of our approach for combinatorial optimization problems. Here the method yields polynomial time approximations and pseudopolynomial, exact algorithms for Unbounded Knapsack Problems. We derive an algorithm for problems with both cost and constraint robustness.

As the general results of this paper could be applied so broadly and quickly already here, we believe they will also be a fruitful method for future research in robust optimization over integers.

* Supported by the Deutsche Forschungsgemeinschaft within the research training group 'Methods for Discrete Structures' (GRK 1408).

** Marie-Curie Fellow of the European Commission under the ROSES-Project (FP7-PEOPLE-2009-IOF 254402).

1 Introduction

A solution to an optimization problem often has to be good not just for one instance but for a set of scenarios. This can either be due to uncertainty as to which of the scenarios will eventually occur, or because the solution shall be used several times in different scenarios.

One solution concept for optimization over scenarios is Robust Optimization. In the robust paradigm feasibility and cost of a solution is measured by those scenarios in which the solution performs worst. This worst case approach contrasts to stochastic programming, where the cost of a solution is typically a weighted average over all scenarios, good ones and bad ones.

As an illustration, suppose we choose a route for regularly driving to work. It makes sense to use some type of average as a measure for fuel consumption in this case. But as far as safety is concerned, one will consider each route according to the risk in its worst case scenario.

Robust optimization has thriven in the past decade, partly because its applicability became apparent, partly because the resulting mathematical models allow for strong solution methods. For continuous problems a cohesive body of quite general methods has been developed. For combinatorial problems and integer linear programs the picture is a lot more scattered. Typically, the results cover a specific combinatorial problem. This is of course a consequence of the richness of combinatorial optimization and integer linear programming. General results for all of these problems as in the continuous case are unlikely. Therefore the following result by Bertsimas and Sim is even more remarkable:

In [7], they show that

- for uncertain cost coefficients, where at most Γ of them can deviate from the nominal setting at the same time, solvability or approximability of any problem with binary decision variables extends to the robust case, as it suffices to solve a linear number of instances of the deterministic problem.

Unfortunately, this result is intrinsically limited to binary variables. We give a corresponding result for integer, not-necessarily binary cost robust programs. Further, we can extend our method to general robust integer programs with uncertainty in one (or few) constraint(s). Restricting this result again to binary problems gives the exact sibling of the cost robust result in [7] for robustness in constraints.

Our Contribution: The main results of this paper are the following:

- The cost robust counterpart (in the same sense as in [7]) of an integer problem can be solved or approximated if the original problem can be solved for piecewise linear convex cost functions with at most two bends.
- To solve integer problems with uncertainty in a constant number of linear constraints, one has to solve a modified problem where the left hand sides of the constraints are replaced by piecewise linear convex functions.
- For *binary* problems with uncertainty in a constant number of linear constraints, it suffices to solve a polynomial number of instances of the original problem with slightly modified coefficients in the constraints.

These general results allow us to develop methods for cost robust counterparts of entire classes of integer linear programs, notably, totally unimodular programs (TUM) and integer programs with two variables per constraint (IP2). Both classes have been studied intensely in the deterministic case, but we are not aware of any general results on their robust counterparts. Our general result on cost robust TUM problems broadly extends results on Robust Min Cost Flows in [7]. Further, we apply our general results to a combinatorial problem, namely, Unbounded Knapsack. In this case we derive an algorithm that handles cost robustness and robustness in the constraints at the same time.

Our result on integer programs with uncertainty in the constraints is limited to a constant number of disturbed constraints. A path breaking work [8] on robust linear programs (and other continuous optimization classes) gives a method to *solve* robust LPs where potentially every constraints is subject to disturbances. This paper [8] has established the role of Γ scenario sets in robust optimization, i.e., the set of all scenarios where each input parameter can vary in its own interval, but the absolute, normalized sum of all variations is limited by a constant Γ. The major part of [8] is devoted to show that these sets imply a good probability for a robust solutions to be feasible outside the assumed scenarios set. This result holds *only* for small, constant

numbers of constraints being subject to uncertainty. Also, primary examples in [8] feature uncertainty in one constraint only. Hence, the loss in solvability for the integer case seems not very important, as already the scenario sets are not well justified for the general case.

Often optimization problems with a natural, non-binary, integer linear programming description can be reformulated as binary integer linear programs. Even granted the incurred blow-up of the instance, this does usually not yield a workaround to apply results for robust binary IPs to the naturally non-binary problem. The hindrance is usually that the scenarios sets of the robust counterparts make no sense once the problem is transformed into an unnatural binary program.

Related Work: To the best of our knowledge we give the first results on cost robust counterparts of general integer programs.

Modern continuous robust optimization started with [21] for convex uncertainty sets and [4,5,6] for ellipsoidal uncertainty sets. Still a good overview for the state of the art is [1]. The Γ-scenario setting from [7,8] has found frequent application. It has also been applied to certain covering problems in a two-stage robust setting [10,15].

Robust Knapsack has so far only been considered in the binary setting. Yu [23] considered the Knapsack Problem with general scenarios for uncertain costs and showed that unless the number of scenarios is constant, the problem is strongly NP-hard. Aissi et al. [2] remarked that Yu's proof already shows that there is no approximation algorithm for this problem at all. For Γ-scenarios, however, the result from [7] applies and gives an FPTAS. Klopfenstein and Nace [16,17] considered polyhedral aspects of the robust Knapsack Problem, and in the context of the chance-constraint version also a weight robust Knapsack Problem. For deterministic Unbounded Knapsack Hochbaum [11] considered a non-linear extension of the problem with concave and convex functions for the cost and weight of the items, respectively. This problem is similar to some of the modified Knapsack problems we use.

Integer linear programs with two variables per inequalities and positive objective function coefficients are a generalization of the minimum weight vertex cover and the minimum weight 2-satisfiability problem (2SAT). Hochbaum et al. [12] and Bar-Yehuda and Rawitz [3] provide a pseudopolynomial time 2-approximation algorithm for this class of integer programs. In case the inequalities are restricted to be monotone, Hochbaum and Naor [13] and Bar-Yehuda and Rawitz [3] provide a pseudopolynomial time exact algorithm. All algorithms explicitly assume that the variables are bounded.

Our results for totally unimodular integer programs generalize results on specific totally unimodular problems, e.g., Min Cost Flows [7].

Structure of the paper. The remainder of this paper is organized as follows: In Section 2 we present the general results on cost robust integer problems, and apply them to problems with a total unimodular description (Section 2.1) and Integer Programs with two variables per inequality (Section 2.2). Section 3 is devoted to the general result on constraint robustness, which is then applied, together with the results from Section 2, to the Unbounded Knapsack Problem (Section 3.1).

2 Uncertainty in the Objective

We start with the general result on cost robust, not necessarily binary problems. We will use $[n]$ for the set $\{1,\dots,n\}$ and $\mathbb{N}$ for the non-negative integers (including 0) here and throughout. By $\mathrm{T}(\mathcal{A})$ we denote the running time of an algorithm $\mathcal{A}$. The formal definition for the considered class of problems reads as follows:

Definition 1 (Cost Robust Optimization Problem). *For the optimization problems* $\min_{x\in X}\{c^{\mathrm{T}}x\}$ *and* $\max_{x\in X}\{c^{\mathrm{T}}x\}$*, given by* $P=(c,X)$ *where* $X\subseteq\mathbb{Z}^n$ *and* $c\in\mathbb{R}^n$*, and a non-negative integer vector* $d\in\mathbb{N}^n$ *together with* $\Gamma\in[n]$*, the* minimization (maximization) (d,Γ)–Cost Robust Counterpart (CRC) of P *is defined by*

$$\min_{x\in X}\left\{c^{\mathrm{T}}x+\max_{\substack{S\subseteq[n]\\|S|\le\Gamma}}\sum_{j\in S}|d_jx_j|\right\}\qquad\text{and}\qquad\max_{x\in X}\left\{c^{\mathrm{T}}x-\max_{\substack{S\subseteq[n]\\|S|\le\Gamma}}\sum_{j\in S}|d_jx_j|\right\}\ , \tag{2.1}$$

respectively.

Our main goal is to show that one can solve or approximate the CRC of P, if one can solve or approximate the following variant of P:

Definition 2 (Modified Optimization Problem). *For the minimization (maximization) problem given by $P = (X, c)$, a vector $c' \in \mathbb{R}^n_{\geq 0}$ and a number $\alpha \geq 0$, the* (c', α)–Modified Minimization (Maximization) Problem (MMin, MMax) *of P is*

$$\min_{x \in X} \left\{ \sum_{j \in [n]} \widetilde{c}_j(x_j) \right\} \qquad \text{and} \qquad \max_{x \in X} \left\{ \sum_{j \in [n]} \widetilde{c}_j(x_j) \right\} , \tag{2.2}$$

respectively, where $\widetilde{c}_j(x) := c_j x + \max\{c'_j x - \alpha, 0\} + \max\{-c'_j x - \alpha, 0\}$ for minimization, $\widetilde{c}_j(x) := c_j x - \max\{c'_j x - \alpha, 0\} - \max\{-c'_j x - \alpha, 0\}$ for maximization problems.

At this point minimization and maximization are fully symmetric. At a later stage it will come in handy to have them defined separately.

Theorem 3. *Consider the optimization problem of $P = (c, X)$ with $X \subseteq \mathbb{Z}^n$ and $c \in \mathbb{R}^n$. Suppose for some $\rho \geq 1$ there is a ρ–approximation algorithm $\mathcal{A}_1$ for the (c', α)–MMin (MMax) of P and arbitrary c' and α. Further suppose, for given $d \in \mathbb{N}^n$ and $\Gamma \in [n]$ there is an algorithm $\mathcal{A}_2$ that computes upper bounds u_j on the absolute value of each variable x_j in the optimal solution of the (d, Γ)–CRC of P.*

Then there is a ρ–approximation algorithm $\mathcal{A}$ for the (d, Γ)–CRC of P with running time $\mathrm{T}(\mathcal{A}) \in \mathcal{O}\big(\mathrm{T}(\mathcal{A}_2) + \max_j\{u_j d_j\} \cdot \mathrm{T}(\mathcal{A}_1)\big)$.

Note that for $\rho = 1$, i.e., if we have an exact algorithm for the modified problem, we can solve the CRC exactly.

Proof. We only consider minimization problems, since we can transform any maximization problem into a minimization problem by taking the negative of the costs. We formulate the inner maximization problem of (2.1) as an integer program (IP). This IP is totally unimodular, thus we can substitute it by its dual LP-relaxation. Then the CRC reads as follows:

$$\begin{cases} \min c^{\mathrm{T}} x + \Gamma\vartheta + \sum_{j \in [n]} (y_j + z_j) & \\ \text{s.t.} \quad y_j + \vartheta \geq d_j x_j \;\forall\, j \in [n] \\ \qquad\quad z_j + \vartheta \geq -d_j x_j \;\forall\, j \in [n] \\ \qquad\quad y, z, \vartheta \geq 0 \\ \qquad\quad x \in X\ . \end{cases} \tag{2.3}$$

In an optimum, $y_j = \max\{d_j x_j - \vartheta, 0\}$ and $z_j = \max\{-d_j x_j - \vartheta, 0\}$, so (2.3) is equivalent to

$$\min_{x \in X, \vartheta \geq 0} \left\{ c^{\mathrm{T}} x + \Gamma\vartheta + \sum_{j \in [n]} \big(\max\{d_j x_j - \vartheta, 0\} + \max\{-d_j x_j - \vartheta, 0\} \big) \right\} . \tag{2.4}$$

For a fixed ϑ, this is equivalent to the (d_j, ϑ)–MMin of P. By the conditions of the Theorem, we can compute a ρ-approximate solution to this problem.

In an optimum we have $|x_j| \leq u_j$, so if $\vartheta \geq \max_j u_j d_j =: \overline{\vartheta}$, for all j both maxima vanish. Hence, if we increase ϑ beyond this number, the objective value increases. Therefore $\overline{\vartheta}$ is an upper bound for the optimal value of ϑ.

Also, the optimal ϑ is integral: Denote by

$$C^*(\vartheta) := \Gamma\vartheta + \min_{x \in X} \left\{ \sum_{j \in [n]} c_j x_j + \max\{d_j x_j - \vartheta, 0\} + \max\{-d_j x_j - \vartheta, 0\} \right\} \tag{2.5}$$

the optimal cost for a fixed ϑ. Since x and d are integral, this function is linear in ϑ within each interval $[k, k+1], k \in \mathbb{N}$. In such an interval the local maximum is obtained for $\vartheta = k$ or for $\vartheta = k + 1$, and thus the global maximum is obtained for some integral ϑ.

We compute all corresponding ρ-approximate solutions and choose the best among them, resulting in the claimed running time. □

Remark. Our model of robustness limits to deviation in at most Γ cost coefficients. The resulting inner maximization problem, which we dualized in the previous proof, is totally unimodular. Therefore a standard argument originating from [7] gives that this model is equivalent to protecting against any cost function $c + \delta d$ with δ in the set $\{\delta \in \mathbb{R}^n : \sum_{j\in[n]} |\delta_j| \leq \Gamma\}$.

Unless $\max_j u_j d_j$ is polynomial in the input, in Theorem 3 one ends up with a pseudopolynomial algorithm for the CRC, even if one has a polynomial algorithm for the modified optimization problem. This can be overcome if $\rho = 1$ and C^* as defined in (2.5) is convex as a function of ϑ, in which case the correct ϑ^* can be found via a carefully constructed binary search (similar to the one in proof of Theorem 7 in [7]).

Theorem 4. *Consider the minimization problem of $P = (c, X)$ with $X \subseteq \mathbb{N}^n$ and $c \in \mathbb{R}^n_{\geq 0}$. Under the conditions of Theorem 3, if $\rho = 1$ and C^* is a convex function, then there is an exact algorithm $\mathcal{A}$ for the (d, Γ)-CRC of P with running time $\mathrm{T}(\mathcal{A}) \in \mathcal{O}\big(\mathrm{T}(\mathcal{A}_2) + \max_j \log(u_j d_j) \cdot \mathrm{T}(\mathcal{A}_1)\big)$.*

For a detailed description of an application of this result we refer the reader to section 2.1.

When $\rho > 1$ or C^* is not convex, we can still restrict the number of calls of the oracle $\mathcal{A}_1$ to $\mathcal{O}(\max_j \log(u_j d_j))$ in exchange for a slightly weaker approximation guarantee. But for this result we have to consider minimization and maximization separately and restrict to combinatorial problems with non-negative cost coefficients and variables. Note that in this case the second maximum in both the definition of MMin and MMax vanishes.

Theorem 5 (Minimization Problem). *Consider the minimization problem of $P = (c, X)$ with $X \subseteq \mathbb{N}^n$ and $c \in \mathbb{R}^n_{\geq 0}$. Under the conditions of Theorem 3, for all $\varepsilon > 0$ there is a $(\rho + \varepsilon)$-approximation algorithm $\mathcal{A}$ for the (d, Γ)-CRC of P with running time $\mathrm{T}(\mathcal{A}) \in \mathcal{O}\big(\mathrm{T}(\mathcal{A}_2) + \frac{1}{\varepsilon} \cdot \max_j \log(u_j d_j) \cdot \mathrm{T}(\mathcal{A}_1)\big)$.*

Proof. We start as in the proof of Theorem 3. To attain the claimed running time, however, for any given $\varepsilon > 0$, we now solve (2.5) approximately for all $\vartheta \in \{0\} \cup \{(1+\varepsilon)^k : k \in \mathbb{N}, (1+\varepsilon)^{k-1} \leq \overline{\vartheta}\}$, and return the best of all these solutions. This yields a $(\rho + \varepsilon)$-approximation for the CRC:

Denote the optimal value of ϑ by ϑ^*. In case $\vartheta^* \in \{0, 1\}$, our solution is within a factor of ρ of the optimum, since these two values for ϑ are checked. Otherwise, let $k_0 \in \mathbb{N} \setminus \{0\}$ be such that $(1+\varepsilon)^{k_0-1} < \vartheta^* \leq (1+\varepsilon)^{k_0} =: \vartheta_0$. Since Γ, ϑ^*, c, and $x \geq 0$, we get[4]

$$\frac{C^*(\vartheta_0)}{C^*(\vartheta^*)} \leq \max\left\{ \frac{\Gamma\vartheta_0}{\Gamma\vartheta^*}, \frac{\min_{x\in X}\left\{\sum_j c_j x_j + \max\{d_j x_j - \vartheta_0, 0\}\right\}}{\min_{x\in X}\left\{\sum_j c_j x_j + \max\{d_j x_j - \vartheta^*, 0\}\right\}} \right\} \leq 1 + \varepsilon \,.$$

Since we can compute ρ-approximations to $C^*(\vartheta)$, the best solution we find is a $(\rho+\varepsilon)$-approximation for the CRC. Further, the oracle $\mathcal{A}_1$ is called $\mathcal{O}(log_{(1+\varepsilon)}\overline{\vartheta}) = \mathcal{O}\big(\frac{1}{\varepsilon} \cdot \max_j \log(u_j d_j)\big)$ times, resulting in the claimed running time. □

For maximization the perturbed cost in a worst scenario can be relatively close to zero, while all numbers involved are rather large. This, roughly speaking, spoils an approximation result for maximization similar to Theorem 5—unless we impose a further condition:

Theorem 6 (Maximization Problems). *Consider the maximization problem of $P = (c, X)$ with $X \subseteq \mathbb{N}^n$ and $c \in \mathbb{R}^n_{\geq 0}$. Suppose the conditions of Theorem 3 hold, and suppose that the relative cost decrease in the (d, Γ)-CRC of P is bounded from below by a positive constant β, i.e.:*

$$\exists\, \beta > 0 : \qquad \frac{(c_j - d_j)}{c_j} \geq \beta \quad \forall\, j \in [n] \,.$$

Then there is a 2ρ-approximation algorithm $\mathcal{A}$ for the (d, Γ)-CRC of P with running time $\mathrm{T}(\mathcal{A}) \in \mathcal{O}\big(\mathrm{T}(\mathcal{A}_2) + \max_j \log(u_j d_j) \cdot \mathrm{T}(\mathcal{A}_1)\big)$.

[4] Using $\frac{a+b}{c+d} \leq \max\left\{\frac{a}{c}, \frac{b}{d}\right\}$ $\forall\, c, d > 0$.

Proof. As in the proof of Theorem 5, we solve the MMax of P for $\vartheta = (1+\varepsilon)^k$ for some $k \in \mathbb{N}$ and a particular $\varepsilon > 0$. For the choice of ε, consider an optimal solution (x^*, ϑ^*) with value OPT. We get that

$$\Gamma\vartheta^* \leq \Gamma\vartheta^* + \sum_{j\in[n]} \max\{d_j x_j^* - \vartheta^*, 0\} = \underbrace{c^{\mathrm{T}}x^* - \mathrm{OPT}}_{(1)} \overset{(*)}{\leq} d^{\mathrm{T}}x^* \leq (1-\beta)c^{\mathrm{T}}x^* \,, \tag{2.6}$$

where $(*)$ holds because (1) is the cost we loose due to the decrease of some of the coefficients, and this cost is bounded by $d^{\mathrm{T}}x^*$.

We now set $\varepsilon := \beta/2(1-\beta)$ (w.l.o.g. $\beta > 1$). Then

$$\mathrm{OPT} \geq (c-d)^{\mathrm{T}}x^* \geq \beta c^{\mathrm{T}}x^* = 2\varepsilon(1-\beta)c^{\mathrm{T}}x^* \overset{(2.6)}{\geq} 2\varepsilon\Gamma\vartheta^* \,. \tag{2.7}$$

With this, we can bound the error that arises from approximating ϑ:

Denote by $C^*(\vartheta) := -\Gamma\vartheta + \max_{x\in X}\left\{\sum_{j\in[n]} c_j x_j - \max\{d_j x_j - \vartheta, 0\}\right\}$ the optimal cost for a fixed ϑ. With ϑ_0 as in the proof of Theorem 5 we then get

$$\begin{aligned}
\frac{\mathrm{OPT}}{C^*(\vartheta_0)} &\leq \frac{\mathrm{OPT}}{-\Gamma\vartheta_0 + \max_{x\in X}\left\{\sum_{j\in[n]}\left(c_j x_j - \max\{d_j x_j - \vartheta^*, 0\}\right)\right\}} && \text{(since } \vartheta_0 \geq \vartheta^*\text{)} \\
&= \frac{\mathrm{OPT}}{-\Gamma\vartheta_0 + \mathrm{OPT} + \Gamma\vartheta^*} && \text{(since } \vartheta^* \text{ optimal)} \\
&\leq \frac{\mathrm{OPT} - \varepsilon\Gamma\vartheta^* + \varepsilon\Gamma\vartheta^*}{-\Gamma(1+\varepsilon)\vartheta^* + \mathrm{OPT} + \Gamma\vartheta^*} && \text{(since } \vartheta_0 \leq (1+\varepsilon)\vartheta^*\text{)} \\
&= 1 + \frac{\varepsilon\Gamma\vartheta^*}{\mathrm{OPT} - \varepsilon\Gamma\vartheta^*} \overset{(2.7)}{\leq} 2 \,.
\end{aligned}$$

Since we are able to approximate the optimal solution to the MMax of P within a factor of ρ, the considerations above prove that our algorithm yields a 2ρ-approximation.

The number of calls of $\mathcal{A}_1$ is the same as in the proof of Theorem 5. Since ε is constant, we get the claimed overall running time. □

2.1 Problems with Totally Unimodular Description

The concept of totally unimodular matrices is arguably the most successful concept for solving a large class of integer programs. In general, robust counterparts need not inherit total unimodularity. We show that in general the CRC of P can be solved exactly for those problems where the solution space of P can be described by a totally unimodular matrix of size polynomial in the size of the input of P.

This generalizes results on specific totally unimodular problems. In particular, it broadly generalizes the results on Robust Network Flows in [7], since the Min Cost Flow Problem is totally unimodular.

In this section we do not require non-negativity of the cost vector, so the minimization results we show can be used for maximization problems as well. The structure of the problems considered here is as follows:

Definition 7. *A minimization problem $P = (c, X)$ is said to have a* bounded TUM description *(A, b, u) if the set of feasible solutions $X \subseteq \mathbb{N}^n$ is described by a totally unimodular matrix $A \in \mathbb{R}^{m\times n}$, an integral right-hand-side b, and an integral vector of upper bounds u, i.e.*

$$\mathrm{conv}(X) = \{x \in \mathbb{R}^n : Ax \leq b, x \leq u\}, \qquad A \text{ TUM}, b, u \in \mathbb{Z}^n.$$

Remark. The non-negativity condition can be lifted. Still this yields much less readable results that rest on similar arguments.

For totally unimodular problems we first show that we can always solve MMin:

Lemma 8. *If the minimization problem $P = (c, X)$ is given by a bounded TUM description (A, b, u), then the Modified Minimization Problem can be solved in polynomial time.*

Proof. Let $\widetilde{c}$ be an arbitrary vector of piecewise linear cost functions as defined in Definition 2. Note that since $x \geq 0$, the second maximum in the cost functions vanishes and they have a single bend at α/c'_j.

We split up each variable into three new variables with slightly different cost coefficients. Their upper bounds are chosen such that their sum corresponds to the original variable. Set $\overline{x}_j := \lceil \alpha/c'_j \rceil$. Bounds and costs are set as follows:

Variable	Upper Bound	Cost Coefficient
$x_j^{(1)}$	$\overline{x}_j - 1$	$c_j^{(1)} := c_j$
$x_j^{(2)}$	1	$c_j^{(2)} := c_j + c'_j \overline{x}_j - \alpha$
$x_j^{(3)}$	$u_j - \overline{x}_j$	$c_j^{(3)} := c_j + c'_j$

Let $\mathbb{1}$ denote the vector of ones of compatible size. The resulting problem then reads:

$$\begin{cases} \min \sum_{j\in[n]} c_j^{(1)} x_j^{(1)} + c_j^{(2)} x_j^{(2)} + c_j^{(3)} x_j^{(3)} \\ \text{s.t.} \quad x^{(1)} + x^{(2)} + x^{(3)} \in X \\ \qquad x^{(1)} \leq \overline{x} - \mathbb{1} \\ \qquad x^{(2)} \leq \mathbb{1} \\ \qquad x^{(3)} \leq u - \overline{x}\ . \end{cases} \tag{2.8}$$

Since $c_j^{(1)} \leq c_j^{(2)} \leq c_j^{(3)}$, it is equivalent to the Modified Minimization Problem (2.2).

On the other hand we have

$$\left.\begin{aligned} x^{(1)} + x^{(2)} + x^{(3)} &\in \operatorname{conv}(X) \\ x^{(1)} &\leq \overline{x} - \mathbb{1} \\ x^{(2)} &\leq \mathbb{1} \\ x^{(3)} &\leq u - \overline{x} \end{aligned}\right\} \iff \underbrace{\begin{bmatrix} A & A & A \\ I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}}_{=:A'} \underbrace{\begin{bmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{bmatrix}}_{=:x'} \leq \underbrace{\begin{bmatrix} b \\ \overline{x} - \mathbb{1} \\ \mathbb{1} \\ u - \overline{x} \end{bmatrix}}_{=:b'}$$

where I is the n-dimensional identity matrix.

The matrix A' is again totally unimodular, and the right-hand-side b' is obviously integral. Therefore the optimum of

$$\begin{cases} \min \sum_{j\in[n]} c_j^{(1)} x_j^{(1)} + c_j^{(2)} x_j^{(2)} + c_j^{(3)} x_j^{(3)} \\ \text{s.t.} \quad x^{(1)} + x^{(2)} + x^{(3)} \in \operatorname{conv}(X) \\ \qquad x^{(1)} \leq \overline{x} - \mathbb{1} \\ \qquad x^{(2)} \leq \mathbb{1} \\ \qquad x^{(3)} \leq u - \overline{x} \end{cases} \tag{2.9}$$

is integral and thus an optimal solution of (2.8). Also, it can be computed in polynomial time. □

Further, in this setting C^* is convex:

Lemma 9. *Let $C^*(\vartheta)$ be defined as in (2.5). Then for a minimization problem $P = (c, X)$ with a bounded TUM description (A, b, u), C^* is convex for any $c \in \mathbb{R}^n, d \in \mathbb{N}^n, \Gamma \in [n]$.*

Proof. Consider problem (2.3), where because $x \geq 0$ the variables z_j and corresponding constraints are removed, and let $(x^{(1)}, y^{(1)})$ and $(x^{(2)}, y^{(2)})$ be optimal solutions to this problem for $\vartheta = \vartheta_1$ and $\vartheta = \vartheta_2$, respectively. Then for arbitrary $\lambda \in [0, 1]$, $(\lambda x^{(1)} + (1-\lambda)x^{(2)}, \lambda y^{(1)} + (1-\lambda)y^{(2)})$ is a feasible solution for $\vartheta = \lambda\vartheta_1 + (1-\lambda)\vartheta_2$, if in the last constraint of (2.3) we replace X by $\operatorname{conv}(X)$.

The proof of Lemma 8 tells us that the optimal solution of this problem is always integral. Hence we get

$$\begin{aligned} &C^*(\lambda\vartheta_1 + (1-\lambda)\vartheta_2) \\ &\quad \leq c^{\mathrm{T}}(\lambda x^{(1)} + (1-\lambda)x^{(2)}) + \Gamma(\lambda\vartheta_1 + (1-\lambda)\vartheta_2) + \mathbb{1}^{\mathrm{T}}(\lambda y^{(1)} + (1-\lambda)y^{(2)}) \\ &\quad = \lambda C^*(\vartheta_1) + (1-\lambda)C^*(\vartheta_2)\ . \end{aligned}$$

This proves that C^* is indeed convex. □

We can now apply Theorem 4 and get that we can solve the CRC of any problem with a bounded totally unimodular description in polynomial time:

Theorem 10. *If the minimization problem $P = (c, X)$ is given by a bounded TUM description (A, b, u), then for any $c \in \mathbb{R}^n, d \in \mathbb{N}^n$ and $\Gamma \in [n]$, there is an exact algorithm for the (d, Γ)–CRC of P that runs in polynomial time.*

2.2 Integer Programs with Two Variables per Inequality

We now apply our main results to a second, large, and intensely studied class of integer programs, namely integer programs with two variables per inequality (IP2).

Definition 11 (Integer Programs with Two Variables per Inequality). *A* bounded integer program with two variables per inequality *(bounded IP2) is a system of the form*

$$\min \{c^{\mathrm{T}} x : a_i^{\mathrm{T}} x \geq b_i \text{ for } i = 1, \ldots, m,\ \ell \leq x \leq u,\ x \text{ integer}\},$$

where $b \in \mathbb{Z}^m$, $\ell, u \in \mathbb{Z}^n$, $c \in \mathbb{Q}^n$ and each vector $a_i \in \mathbb{Z}^n$ has two non-zero components.

A bounded IP2 is called monotone *if the non-zero coefficients of a_i have opposite signs.*

The conditions required in Section 2 allow to intensely use the existing techniques for non-robust IP2, in particular [12], [13] and [3]. We obtain a pseudopolynomial time 2–approximation for the CRC of bounded IP2s and an exact pseudopolynomial time algorithm for the CRC of bounded, monotone IP2s.

Theorem 12. *The cost robust counterpart of a bounded monotone IP2 can be solved in pseudopolynomial time.*

Remark. In the appendix, we prove Theorem 12 by extending (to handle piecewise linear functions) the pseudopolynomial time algorithm of Hochbaum and Naor [13] for bounded monotone IP2. In [3], Bar-Yehuda and Rawitz give an exact pseudopolynomial algorithm for monotone cost functions but non-negative lower bounds.

Theorem 13. *There is a pseudopolynomial time 2-approximation algorithm for the cost robust counterpart of a bounded IP2 with non-negative coefficients in the objective function and non-negative lower bounds.*

Remark. In the appendix, we prove 13 by extending (to handle piecewise linear functions) the pseudopolynomial 2-approximation algorithm of Hochbaum et al. [12]. This result is also shown in [3].

3 Uncertainty in Constraints

We now turn to the case where the coefficients of a single linear constraint (or those of a constant number of them) are uncertain. All results presented here hold both for minimization and maximization, so we restrict to one of the two. The formal definition of the considered class of problems is as follows:

Definition 14 (Constraint Robust Maximization Problem). *Consider the problem $\max_{x \in X}\{c^{\mathrm{T}} x\}$, given by $P = (c, X)$ where $c \in \mathbb{R}^n$ and $X = \{x \in X' : a^{\mathrm{T}} x \leq r\}$ for some $X' \subseteq \mathbb{Z}^n, a \in \mathbb{R}^n, r \in \mathbb{R}$. For a non-negative integer vector $b \in \mathbb{N}^n$ together with $\Gamma \in [n]$, the* (b, Γ)–Constraint Robust Counterpart (ConsRC) of P *is defined by*

$$\max c^{\mathrm{T}} x \qquad s.t. \quad x \in X', \quad a^{\mathrm{T}} x + \max_{\substack{S \subseteq [n] \\ |S| \leq \Gamma}} \sum_{j \in S} |b_j x_j| \leq r. \tag{3.1}$$

As in the cost robust setting, the left hand side of the constraint with uncertain coefficients can be transformed into a sum of piecewise linear convex function with two bends:

Lemma 15. *The (b,Γ)-Constraint Robust Counterpart of the maximization problem $P=(c,X)$ as defined in Definition 14 is equivalent to*

$$\max_{\xi\geq 0}\max_{x\in X(\xi)} c^{\mathrm{T}}x \tag{3.2}$$
$$\text{with}\quad X(\xi):=\left\{x\in X' : \Gamma\xi+\sum_{j\in[n]} a_jx_j+\max\{b_jx_j-\xi,0\}+\max\{-b_jx_j-\xi,0\}\leq r\right\}.$$

Proof. With the same transformations as in the cost robust setting, we get that (3.1) is equivalent to

$$\max c^{\mathrm{T}}x$$
$$\text{s.t.}\quad x\in X',\quad a^{\mathrm{T}}x+\min_{\xi\geq 0}\left\{\Gamma\xi+\sum_{j\in[n]} a_jx_j+\max\{b_jx_j-\xi,0\}+\max\{-b_jx_j-\xi,0\}\right\}\leq r.$$

Thus, for all feasible solutions x of (3.1) there exists some $\xi(x)\geq 0$ such that $x\in X(\xi(x))$. Consequently, as claimed (3.1) is equivalent to $\max_{\xi\geq 0}\max_{x\in X(\xi)} c^{\mathrm{T}}x$. □

For the non-binary case, the optimal value ξ^* can be found by enumeration, since it is integral and bounded by the maximum deviation in the constraint coefficients.

Corollary 16. *Consider the (b,Γ)-ConsRC of the maximization problem $P=(c,X)$ as defined in Definition 14. Suppose there is an algorithm $\mathcal{A}_1$ computing a ρ-approximation for $\max_{x\in X(\xi)} c^{\mathrm{T}}x$ for any $\xi\geq 0$, and an algorithm $\mathcal{A}_2$ that computes upper bounds u_j on the absolute value of each variable x_j in the optimal solution of (3.1). Then there is a ρ-approximation algorithm $\mathcal{A}$ for the (b,Γ)-ConsRC of P with running time $\mathrm{T}(\mathcal{A})=\mathcal{O}\big(\mathrm{T}(\mathcal{A}_2)+\max_j\{u_jb_j\}\cdot\mathrm{T}(\mathcal{A}_1)\big)$.*

If all variables are binary, i.e. $X'\subseteq\{0,1\}^n$, there are only $n+1$ possible values for ξ^*, and for a fixed ξ the constraint of problem (3.2) becomes linear again. Hence, to solve the (b,Γ)–ConsRC of $P=(c,X)$, it suffices to solve $n+1$ problems of the type of P for slightly different coefficients in the linear constraint:

Theorem 17. *If $X'\subseteq\{0,1\}^n$, the (b,Γ)-ConsRC of the maximization problem $P=(c,X)$ as defined in Definition 14 is equivalent to*

$$\max_{\ell=1,\dots,n+1}\left(\max c^{\mathrm{T}}x\qquad s.t.\quad x\in X',\quad \Gamma b_\ell+a^{\mathrm{T}}x+\sum_{j=1}^{\ell-1}(b_j-b_\ell)x_j\leq r\right),$$

whereby w.l.o.g. we assume $b_n\leq b_{n-1}\leq\ldots\leq b_1$ and define $b_{n+1}:=0$.

Remark. This result is an exact sibling of the result on cost robust binary problems in [7], but with uncertainty in a constraint instead of the costs.

Proof. The optimal value ξ^* lies in $[0,b_1]$. We split up this interval at $b_\ell,\ell=n\ldots,2$, and maximize over each subinterval, i.e. we reformulate (3.2) to get

$$\max_{\ell=1,\dots,n}\left(\max_{\xi\in[b_{\ell+1},b_\ell]}\left(\max_{x\in X(\xi)} c^{\mathrm{T}}x\right)\right). \tag{3.3}$$

For $x\in\{0,1\}^n$ we have $\max\{b_jx_j-\xi,0\}=\max\{b_j-\xi,0\}x_j$, and thus for $\xi\in[b_{\ell+1},b_\ell]$

$$\begin{aligned}X(\xi)&=\left\{x\in X' : \Gamma\xi+\sum_{j\in[n]} a_jx_j+\max\{b_j-\xi,0\}x_j\leq r\right\}\\&=\left\{x\in X' : \Gamma\xi+a^{\mathrm{T}}x+\sum_{j=1}^{\ell}(b_j-\xi)x_j\leq r\right\}.\end{aligned} \tag{3.4}$$

For any fixed x, the left hand side of the constraint in (3.4) is a linear function in ξ that has to be no greater than r somewhere in $[b_{\ell+1}, b_\ell]$ for x to be feasible. Thus, if the constraint is satisfied for any ξ in this interval, because of linearity it will be satisfied for at least one of the values $\xi = b_{\ell+1}$ or $\xi = b_\ell$. As a consequence,

$$\max_{\xi\in[b_{\ell+1},b_\ell]} \left(\max_{x\in X(\xi)} c^{\mathrm{T}} x \right) = \max_{\xi=b_{\ell+1},b_\ell} \left(\max_{x\in X(\xi)} c^{\mathrm{T}} x \right). \tag{3.5}$$

Combining (3.3)–(3.5) this yields the claimed result. □

Remark. A direct corollary from Theorem 17 is the existence of an FPTAS for the weight robust counterpart of the binary Knapsack Problem, generalizing a result from [17].

All the results from this section hold as well if there is a constant number k of constraints with uncertain coefficients. The problem $\max_{x\in X(\xi)}$ would then have to be solved $(\max_j\{u_j b_j\})^k$ times in the setting of Corollary 16 and $(n+1)^k$ times in the binary case.

3.1 Robust Unbounded Knapsack Problems

To demonstrate how versatile our main results are for combinatorial problems, we apply them to the *Unbounded Knapsack Problem*, the non-binary extension of the classical Knapsack Problem (KP). For this problem we will be able to handle counterparts that feature both, cost robustness and robustness in the constraint.

To distinguish between the standard version of the Unbounded Knapsack and a technically relevant modification used further below, we speak of the *Linear* and the *Concave* Unbounded Knapsack Problem.

Definition 18 (Linear Unbounded Knapsack Problem). *An instance of the* Linear Unbounded Knapsack Problem *(UKP) is given by a knapsack capacity $W \geq 0$ and n types of items with weights $w_j \in \mathbb{N}$ and costs $c_j \in \mathbb{R}_{\geq 0}$, $j \in [n]$. The task is to find a vector $x \in \mathbb{N}^n$ with $\sum_j w_j x_j \leq W$ maximizing the cost $\sum_j c_j x_j$.*

The linear UKP and its later defined extensions, in particular its robust counterparts, are *NP*-hard, by the weak *NP*-hardness of the binary KP. Intuitively, UKP seems to be more complex than the binary KP, since the input is more compact. Still, as for KP, there is both a pseudopolynomial Dynamic Program (DP) and an FPTAS [18,14].

We now consider the robust versions of the Unbounded Knapsack Problem.

Cost Robust UKP. While the result for binary cost robust programs [7] can be applied to the standard Knapsack Problem, the CRC of the UKP surpasses the reach of [7]. One may argue that the UKP can be formulated as a binary integer program spending a variable for each potentially useful copy of an item. But, apart from the enormous blow up in size, the robust counterpart of such an IP, where at most a fixed number of cost coefficients may vary, is completely different to that of the UKP—and apparently does not make much sense.

To tackle the *Cost Robust Unbounded Knapsack Problem* (CRUKP) with our results from Section 2, we consider the following problem:

Definition 19 (Concave Unbounded Knapsack Problem). *An instance of the* Concave Unbounded Knapsack Problem *(ConcUKP) is given by a knapsack capacity W, weights $w_j \in \mathbb{N}, j \in [n]$, and* monotonically increasing, concave cost functions $c_j : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, $j \in [n]$, *given by an oracle.*

The task is to find a vector $x \in \mathbb{N}^n$ with $\sum_j w_j x_j \leq W$ maximizing the cost $\sum_j c_j(x_j)$.

This problem has been considered in literature before, as it has a lot of applications. Hochbaum presented an FPTAS for it in [11]. We sketch an alternative FPTAS based on a dynamic program (DP). The detailed proofs are moved to the appendix.

To construct the FPTAS, we consider the following DP for ConcUKP: Iteratively compute state spaces $\mathcal{S}_j, j = 1, \ldots, n$. A state $[Y, C] \in \mathcal{S}_j$ represents a partial solution with weight Y and cost C using only items from the subset $\{1, \ldots, j\}$. $\mathcal{S}_j$ is created by adding any feasible number of items of type j to each state in

$\mathcal{S}_{j-1}$, starting with $\mathcal{S}_0 := \{[0,0]\}$. Through elimination of dominated states we can ensure $|\mathcal{S}_j| \leq W+1$ at all times, thus bounding the number of calls to the cost oracle by $\mathcal{O}(n^2W)$.

This DP can be transformed into an FPTAS by a general scheme due to Woeginger [22]. To get polynomial running time, however, it has to be slightly modified: Instead of considering every possible number of items of each type to be added to the knapsack, we only consider powers of $1+\varepsilon$. The concavity of the cost functions ensures that the loss arising through this is bounded by a factor of $1+\varepsilon$, preserving the approximation guarantee of the FPTAS.

Now observe that the cost functions of the Modified Maximization Problem for UKP are feasible cost functions for ConcUKP. Also, $\lfloor W/w_j \rfloor$ is an upper bound on x_j with a polynomial encoding length. Therefore, we can apply Theorem 6 to get the following result:

Theorem 20. *For all $\varepsilon > 0$, there is a $(2+\varepsilon)$–approximation algorithm for CRUKP, if the relative cost decrease is bounded from below by a constant.*

Weight Robust UKP. Next we turn to the Unbounded Knapsack Problem where weights instead of costs are uncertain. At most Γ types of items can increase their weight, and we are looking for a solution that is feasible in all scenarios, maximizing the cost. In terms of section 3, we have uncertainty in the only constraint. Let Δw_j denote the possible increase in weight of items of type j. We consider the $(\Delta w, \Gamma)$-ConsRC of UKP, also called the *Weight Robust Unbounded Knapsack Problem* (WRUKP).

From Corollary 16 we learn that we have to solve $\max_{x \in X(\xi)} c^{\mathrm{T}} x$ in order to get a pseudopolynomial algorithm for WRUKP. The FPTAS from [11] could be used for this, but we will focus on computing an exact algorithm in pseudopolynomial time here. This can be done by the DP presented above for ConcUKP, slightly modified to work with piecewise linear weight functions with a single bend. With $u_j = \frac{W}{w_j}$, we get an algorithm for WRUKP with running time $\mathcal{O}\left(\max \frac{\Delta w_j}{w_j} \cdot n^2 W^2\right)$.

General Robust UKP. Finally, we consider a version of UKP where both weights and costs are uncertain, called the *General Robust Unbounded Knapsack Problem* (RUKP). At most Γ_w types of items can increase their weight, and at most Γ_c cost coefficients decrease. The goal is to find a solution x that is feasible in all scenarios and maximizes the worst-case cost:

$$\max_x \left\{ \sum_{j\in[n]} c_j x_j - \max_{\substack{S_c \subseteq [n] \\ |S_c| \leq \Gamma_c}} \left\{ \sum_{j \in S_c} d_j x_j \right\} \right\} \quad \text{s.t.} \quad \sum_{j\in[n]} w_j x_j + \max_{\substack{S_w \subseteq [n] \\ |S_w| \leq \Gamma_w}} \left\{ \sum_{j \in S_w} \Delta w_j x_j \right\} \leq W \ .$$

This is the $(\Delta w, \Gamma_w)$–ConsRC of CRUKP. We adapt the DP for ConcUKP such that it works for piecewise linear weight functions. By Theorem 3 we get an exact algorithm $\mathcal{A}_1$ for CRUKP on the modified solution space $X(\xi)$ with a running time of $\mathrm{T}(\mathcal{A}_1) = \mathcal{O}\left(\max_j \frac{d_j}{w_j} \cdot n^2 W^2\right)$. Thus we can solve RUKP exactly in a running time of $\mathcal{O}\left(\max_j \frac{\Delta w_j}{w_j} \cdot \max_j \frac{d_j}{w_j} \cdot n^2 W^3\right)$.

Alternatively, if the relative cost decrease is bounded from below by a constant, we can use the $(2+\varepsilon)$–approximation algorithm $\mathcal{A}_1'$ for CRUKP with polynomial running time to get a $(2+\varepsilon)$–approximation for RUKP in time $\mathcal{O}\left(\max_j \frac{\Delta w_j}{w_j} \cdot W \cdot \mathrm{T}(\mathcal{A}_1')\right)$.

Acknowledgement

We are grateful to Martin Skutella and Günter Rote for discussions that substantially enhanced this paper.

References

1. Special issue on robust optimization. *Mathematical Programming B*, 107(1-2), 2006.
2. H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation of min-max and min-max regret versions of some combinatorial optimization problems. *European Journal of Operational Research*, 179(2):281–290, 2007.

3. R. Bar-Yehuda and D. Rawitz. Efficient algorithms for integer programs with two variables per constraint 1. *Algorithmica*, 29(4):595–609, 2001.
4. A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.
5. A. Ben-Tal and A. Nemirovski. Robust solutions to uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.
6. A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000.
7. D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1-3):49–71, 2003.
8. D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.
9. E. Cohen and N. Megiddo. Improved algorithms for linear inequalities with two variables per inequality. In *Proceedings of the 23th Annual ACM symposium on Theory of Computing*, pages 145–155. ACM, 1991.
10. U. Feige, K. Jain, M. Mahdian, and V. Mirrokni. Robust combinatorial optimization with exponential scenarios. In *IPCO '07: Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization*, pages 439–453, Berlin, Heidelberg, 2007. Springer-Verlag.
11. D. Hochbaum. A nonlinear knapsack problem. *Operations Research Letters*, 17(3):103–110, 1995.
12. D. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Mathematical Programming*, 62(1):69–83, 1993.
13. D. Hochbaum and J. Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, 23:1179–1192, 1994.
14. O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of ACM*, 22:463–468, 1975.
15. R. Khandekar, G. Kortsarz, V. Mirrokni, and M. Salavatipour. Two-stage robust network design with exponential scenarios. In *ESA '08: Proceedings of the 16th annual European Symposium on Algorithms*, pages 589–600, Berlin, Heidelberg, 2008. Springer-Verlag.
16. O. Klopfenstein and D. Nace. A note on polyhedral aspects of a robust knapsack problem. Available online at http://www.optimization-online.org/DB_FILE/2006/04/1369.pdf, 2007.
17. O. Klopfenstein and D. Nace. A robust approach to the chance-constrained knapsack problem. *Operations Research Letters*, 36(5):628–632, 2008.
18. S. Martello and P. Toth. *Knapsack Problems. Algorithms and Computer Implementations.* John Wiley and Sons, 1990.
19. N. Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM J. Comput.*, 12(2):347–353, 1983.
20. J. C. Picard. Maximal closure of a graph and applications to combinatorial problems. *Management Science*, 22(11):1268–1272, 1976.
21. A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.
22. G. Woeginger. When does a dynamic programming formulation guarantee the existence of an FPTAS? In *SODA '99: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 820–829, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.
23. G. Yu. On the max-min 0-1 knapsack problem with robust optimization applications. *Operations Research*, 44(2):407–415, 1996.

Appendix

An Algorithm for Bounded Monotone IP2

Theorem 12 (revisited). *The cost robust counterpart of a bounded monotone IP2 can be solved in pseudopolynomial time.*

Proof. In what follows, we require the following graph concept. A *closed set* of a digraph is a subset of nodes with no outgoing arcs to its complement. The *minimum weight closure* for a digraph is the problem of finding a closed set with minimum weight. It was shown to be polynomial-time solvable by Picard [20].

Using Theorem 3, it is enough to solve the modified problem

$$\mathcal{Q}_M = \min\left\{\sum_j \widetilde{c}_j(x_j)\colon a_i^{\mathrm{T}}x \geq b_i \text{ for } i=1,\dots,m,\ \ell \leq x \leq u,\ x \text{ integer}\right\}$$

associated to a bounded monotone IP2 system

$$\mathcal{Q} = \min\{c^{\mathrm{T}}x\colon a_i^{\mathrm{T}}x \geq b_i \text{ for } i=1,\dots,m,\ \ell \leq x \leq u,\ x \text{ integer}\}$$

in pseudopolynomial time. Recall that $\widetilde{c}_j(x_j) = c_jx_j + \max\{d_jx_j - \vartheta, 0\} + \max\{-d_jx_j - \vartheta, 0\}$.

Hochbaum and Naor [13] proposed a pseudopolynomial time algorithm for a bounded monotone IP2 that reduces the problem to the minimum weight closure of a digraph with pseudopolynomially many nodes. They established a bijection between the feasible solutions of $\mathcal{Q}$ and the non-empty closed sets of a certain digraph G which we will also use: The variables $\ell_j \leq x_j \leq u_j$ are represented in G by disjoint directed paths P_j having $u_j - \ell_j + 1$ nodes identified with the numbers from u_j to ℓ_j in consecutive decreasing order. Each inequality $a_{ir}x_r - a_{is}x_s \geq b_i$ of $\mathcal{Q}$ (where $a_{ir}\cdot a_{is} > 0$) is represented in G by a collection of arcs from P_s to P_r: the node p of path P_s is connected to the node $\lceil (a_{is}p + b_i)/a_{ir}\rceil$ of path P_r, for every p such that the latter node exists. Finally, there is a special vertex v which is connected, bidirectionally, to the last node ℓ_j of every path P_j. This ensures that any closed set $S \neq \emptyset$ contains v, forcing $P_j \cap S$ to be a collection of nodes with consecutive labels from p_j to ℓ_j, for some $\ell_j \leq p_j \leq u_j$. In the bijection, this is interpreted as the assignment $x_j = p_j$ for variable x_j.

We use this bijection to solve $\mathcal{Q}_M$ (see Algorithm 1). We need to set weights for the nodes of G so that if S_x is the closed set associated to the feasible solution x for $\mathcal{Q}_M$, then the total weight of $P_j \cap S_x$ is equal to $\widetilde{c}_j(x_j)$. We achieve this with the following weights $w_j(p)$ for the nodes $p \in P_j$: Set $w_j(l_j) = c_j\ell_j$ and $w_j(p) = \widetilde{c}_j(p) - \widetilde{c}_j(p-1)$ for $\ell_j + 1 \leq p \leq u_j$. Finally, we find, using Picard's algorithm [20], the minimum weight closure of G with weights $\{w_j(\cdot)\}_j$ and a suitable weight for v that makes the minimum closure non-empty. □

A 2–Approximation Algorithm for Bounded IP2

Theorem 13 (revisited). *There is a pseudopolynomial time 2-approximation algorithm for the cost robust counterpart of a bounded IP2 with non-negative coefficients in the objective function and non-negative lower bounds.*

Proof. We closely follow the results in [12]. Consider the modified minimization problem $\mathcal{S}_M = \min\{\sum \widetilde{c}_j(x_j)\colon a_i^{\mathrm{T}}x \geq b_i,\ \ell \leq x \leq u,\ x \text{ integer}\}$ of a bounded IP2 instance with non-negative coefficients in the objective function and non-negative lower bounds. In this case, we just need to consider $\widetilde{c}_j(x_j) = c_jx_j + \max\{d_jx_j - \vartheta, 0\}$, where c_j and d_j are non-negative numbers. We associate two variables x_j^+ and x_j^- to each variable x_j in $\mathcal{S}_M$ and then define the following monotone system $\mathcal{S}'_M$:

$$\min\ \frac{1}{2}\sum_{j=1}^{n}\left(\widetilde{c}_j(x_j^+) + \widetilde{c}_j(-x_j^-)\right)$$

$$\begin{aligned}
a_{iv}x_v^+ - a_{iw}x_w^- \geq b_i \quad &\text{and} \quad -a_{iv}x_v^- + a_{iw}x_w^+ \geq b_i, \quad \forall\, a_{iv}\cdot a_{iw} > 0\\
a_{iv}x_v^+ + a_{iw}x_w^+ \geq b_i \quad &\text{and} \quad -a_{iv}x_v^- - a_{iw}x_w^- \geq b_i, \quad \forall\, a_{iv}\cdot a_{iw} < 0\\
\ell \leq x^+ \leq u \quad &\qquad -u \leq x^- \leq -\ell\\
x^+, x^- \in \mathbb{Z}^n \quad &
\end{aligned}$$

Algorithm 1: Algorithm for bounded monotone IP2

$V = \{(j,p) : j \in \{1,\dots,n\} \text{ and } \ell_j \le p \le u_j\} \cup \{v\}$
$E^{\text{var}} = \{((j,p),(j,p-1)) : j \in \{1,\dots,n\} \text{ and } \ell_j + 1 \le p \le u_j\}$
$E^{\text{v}} = \{((j,\ell_j),v) : j \in \{1,\dots,n\}\} \cup \{(v,(j,\ell_j)) : j \in \{1,\dots,n\}\}$
forall $a_{ir}x_r - a_{is}x_s \ge b_i$ *of* $\mathcal{Q}$ *where* $a_{ir}, a_{is} > 0$ **do**
    $E_i^{\text{ineq}} = \{((s,p),(r,\lceil (a_{is}p + b_i)/a_{ir} \rceil)) : \ell_s \le p \le u_s\} \cap V \times V$
endfor
$E = E^{\text{var}} \cup E^{\text{v}} \cup \bigcup_i E_i^{\text{ineq}}$
for $j := 1$ **to** n **do**
    $w((j,\ell_j)) = c_j \ell_j$
    $w((j,p)) = \widetilde{c}_j(p) - \widetilde{c}_j(p-1) \quad$ for $\ell_j + 1 \le p \le u_j$
endfor
$w(v) = -n \cdot \max_j\{|c_j|\} \cdot \max_j\{|\ell_j| + |u_j|\}$
Find minimum weight closure S of $G = (V,E)$ using w as weight function
for $j := 1$ **to** n **do**
    $x_j = \max\{p : (j,p) \in V \cap S\}$
endfor
return x

Note that if x is feasible for $\mathcal{S}_M$, then $x^+ = x$, $x^- = -x$ is feasible for $\mathcal{S}'_M$. Moreover, their respective costs coincide and therefore $OPT(\mathcal{S}'_M) \le OPT(\mathcal{S}_M)$. We can compute an optimal solution $(\tilde{x}^+, \tilde{x}^-)$ for $\mathcal{S}'_M$ using the algorithm in Theorem 12 and we can also compute an arbitrary feasible solution z for $\mathcal{S}_M$, using the strongly polynomial time algorithm from [19,9]. Finally, we construct the 2-approximate solution $\widehat{x}$ for $\mathcal{S}_M$ using z and $\tilde{x}$ as follows:

- If z_j lies in the interval between $\tilde{x}_j^+$ and $-\tilde{x}_j^-$, we set $\widehat{x}_j = z_j$.
- If z_j is less than both $\tilde{x}_j^+$ and $-\tilde{x}_j^-$, we set $\widehat{x}_j = \min\{\tilde{x}_j^+, -\tilde{x}_j^-\}$.
- If z_j is greater than both $\tilde{x}_j^+$ and $-\tilde{x}_j^-$, we set $\widehat{x}_j = \max\{\tilde{x}_j^+, -\tilde{x}_j^-\}$.

In [12], they prove that this solution is feasible for $\mathcal{S}_M$. In order to prove the approximation guarantee, we need a introduce a new argument. Note that, since the objective function coefficients and the lower bounds are non-negative, then $\widetilde{c}_j(\widehat{x}_j) \le \widetilde{c}_j(\tilde{x}_j^+) + \widetilde{c}_j(-\tilde{x}_j^-)$. Therefore the value of $\widehat{x}$ in $\mathcal{S}_M$ is at most $2OPT(S'_M) \le 2OPT(\mathcal{S}_M)$. It is easy to check the proposed algorithm runs in pseudopolynomial time. □

An FPTAS for ConcUKP

The details of the DP sketched in Section 3.1 are listed in Algorithm 2. By elimination of dominated states the size of each $\mathcal{S}_j$ can be reduced to at most $W + 1$. Every time a new state is created, we have to call the oracle for the cost function, thus the number of these calls is in $\mathcal{O}(nW^2)$.

As outlined above, the DP can be transformed into an FPTAS with the result from [22].

Lemma 21. *There is an FPTAS for ConcUKP.*

Proof. In Algorithm 2, out of each state we create up to $u_j + 1$ new states, which is not polynomial in the input size as demanded in [22], Condition C.4(ii). Therefore, we modify the DP as follows: For a given accuracy $\varepsilon > 0$, instead of all possible numbers of items, only consider powers of $1+\varepsilon$ in line 6 of Algorithm 2.[5]

To see that the solution of the modified DP is within a factor of $1+\varepsilon$ of the optimum, consider an optimal solution x^* and the corresponding DP solution $\overline{x}$:

$$(1+\varepsilon)^{k_j} \le x_j^* < (1+\varepsilon)^{k_j+1} \qquad \Rightarrow \quad \overline{x}_j = (1+\varepsilon)^{k_j} .$$

[5] To get an integral solution, the powers $(1+\varepsilon)^k$ could be rounded up, preserving the approximation guarantee. To simplify things, however, we consider the unrounded values.

Algorithm 2: DP for ConcUKP

```
u_j := ⌊W/w_j⌋ ∀ j ∈ [n]
S_0 := {[0,0]}
for j := 1 to n do
    S_j := ∅
    forall [Y,C] ∈ S_{j-1} do
        for i := 0 to u_j do
            if Y + i·w_j > W then
                break
            endif
            S_j := S_j ∪ {[Y + i·w_j, C + c_j(i)]}
        endfor
    endfor
endfor
return max{C : [Y,C] ∈ S_n}
```

Since c_j is non-decreasing (i), concave (ii) and non-negative (iii), for all $j \in [n]$ we get

$$c_j(\overline{x}_j) \overset{(i)}{\geq} c_j\left(\frac{1}{1+\varepsilon}x_j^*\right) \overset{(ii)}{\geq} \frac{1}{1+\varepsilon}c(x_j^*) + \left(1 - \frac{1}{1+\varepsilon}\right)c(0) \overset{(iii)}{\geq} \frac{1}{1+\varepsilon}c(x_j^*)\ . \tag{3.6}$$

Denote the best value found by the modified DP by C_{DP} and the optimal cost by OPT. Then it follows that

$$C_{\text{DP}} \geq \sum_{j\in[n]} c_j(\overline{x}_j) \overset{(3.6)}{\geq} \frac{1}{1+\varepsilon} \cdot \sum_{j\in[n]} c_j(x_j^*) = \frac{1}{1+\varepsilon} \cdot \text{OPT}\ .$$

Therefore, the best solution found by the DP is indeed a $(1+\varepsilon)$-approximation.

Now we show that the modified DP satisfies the conditions from [22]. Let $\overline{k} := \max\{k \in \mathbb{N} \setminus \{0\} : (1+\varepsilon)^k \leq \max_j(u_j)\}$. We have a family of mappings $\mathcal{F} = \{\widetilde{F}, F_0, F_1, \ldots, F_{\overline{k}}\}$ creating new states out of a state $[Y, C]$ and an input vector $[w_j, c_j]$, as well as a family $\mathcal{H} = \{\widetilde{H}, H_0, H_1, \ldots, H_{\overline{k}}\}$ with mappings checking the feasibility of the new states. They are defined as follows:

$$\begin{aligned}
\widetilde{F}(w_j, c_j, Y, C) &:= [Y, C] \\
\widetilde{H}(w_j, c_j, Y, C) &:= 0 \\
F_k(w_j, c_j, Y, C) &:= [Y + (1+\varepsilon)^k w_j, C + c_j((1+\varepsilon)^k)] \\
H_k(w_j, c_j, Y, C) &:= Y + (1+\varepsilon)^k w_j - W\ .
\end{aligned}$$

We set the degree-vector $D := [1, 1]$, and two relations on the states as follows:

$$S \preceq_{dom} S' \quad \Leftrightarrow \quad S = S'\ , \qquad\qquad [Y, C] \preceq_{qua} [Y', C'] \quad \Leftrightarrow \quad Y \geq Y'\ .$$

Conditions C.1 through C.3 from [22] can then be verified. With the modification from above, $|\mathcal{F}|$ is linear in the input:

$$|\mathcal{F}| = \overline{k} + 2 \leq \log_{(1+\varepsilon)}(\max_{j\in[n]} u_j) + 2\ .$$

Thus the DP also satisfies C.4 and can therefore be turned into an FPTAS. Since the modified DP is itself a $(1+\varepsilon)$-approximation, we get an FPTAS for ConcUKP. □