Plan B

Uncertainty/Time Trade-Offs for Linear and Integer Programming

Claire Kenyon and Meinolf Sellmann

Brown University, Department of Computer Science
115 Waterman St, Providence, RI 02912, U.S.A.
{claire,sello} @cs.brown.edu

Abstract. We address the following dilemma: When making decisions in real life, we often face the problem that, while we have time to contemplate about a problem, we are not entirely sure what the exact parameters of our problem will be. And, on the other hand, as soon as the real world is revealed to us, we need to act quickly and have no more time to rethink our actions extensively.
We suggest an approach that allows to trade uncertainty for time and marginal quality loss and discuss its applicability to combinatorial optimization problems that can be formulated as linear and integer linear programs. The core idea consists in solving a polynomial number of problems in the extensive time period before the day of operation, so that, as soon as complete information is available, a feasible near-optimal solution to the problem can be found in sublinear time.

1 Introduction

When solving real-world optimization problems, we face a trade-off between time and uncertainty. Often, there is enough time to consider a problem before a solution needs to be realized, but the information available about the problem at that stage is associated with a lot of uncertainty; complete information may be available only immediately before the first decision needs to be made, when almost no time is left anymore to optimize. Consider, for instance, a delivery company that, every night, needs to solve a vehicle routing problem with time windows to schedule the deliveries for the next day in a network with some notoriously unreliable links. In the morning, when the trucks are about to be loaded, it may be much clearer what the delay on the uncertain links is going to be, but waiting 20 minutes to reschedule before loading can begin is just not an option.

Related Work: There has been conducted extensive research that tries to tackle uncertainty. It is impossible to cover this very active research area entirely here, so we need to focus on the work that appears most relevant with respect to our contribution. One of the first big areas to consider unknown future events was online optimization (for an introduction see [4]) where decisions need to be taken already before future events are visible. The worst-case character of adversarial competitive analysis is often perceived as too pessimistic for real world applications, though, since we may want to optimize our expected rewards rather than to shelter ourselves against the worst case.

This motivation combined with the assumption that stochastic information about future events may be available beforehand has lead to the paradigm of stochastic optimization (for an overview on stochastic programming see for example [3]). The standard scenario considered here is a two-stage process where actions can be taken in stage one at a known cost that must then be augmented to a feasible plan by taking additional actions in stage two for costs that were unknown in stage one but that are drawn from a known probability distribution.

A paradigm that treats uncertainty where all decisions need to be taken before the real parameters are revealed to us is robust optimization (see for example [2]). In robust optimization, a solution needs to be computed that is required to be feasible with respect to all possible future scenarios. It is of course appealing to have one solution that will work no matter what, and in some critical applications this appears necessary. However, the loss in solution quality when comparing against the quality that could have been achieved in the scenario that we really face eventually can be enormous.

Therefore, it has been suggested to drop the requirement that just one solution must satisfy all future scenarios. This approach is very natural: In critical circumstances, humans often do not only make one plan for the future, but instead prepare one or several back-up plans that can kick in when the original one turns out to be infeasible. So we may also decide to compute more than one solution in order to be prepared for changing future constraints.

The classic example for this approach is the k-shortest path problem where the k best solutions to the problem are computed (see [7] for an overview of literature). The problem is of course interesting in and by itself. With respect to robustness, however, it is easy to construct examples where this approach fails miserably when just one edge is not available when it comes to executing the shortest path, because all k best solutions relied on the existence of that one edge.

Consequently, some approaches try to compute solutions that differ a lot, in the hope that different solutions are less likely to fail for the same change in the requirements. One example for this approach is the so-called penalty method [12] where an additional cost is introduced that penalizes similarity to an existing solution (whereby it is unclear how to reasonably quantify the trade-off between solution quality and similarity). When restricting ourselves to pure feasibility problems, another example is the search for so-called (a, b)-super solutions in constraint programming [9]. Here, solutions are sought-after that are robust in the sense that, if a variables lose their values, then the solution can be repaired by reassigning the critical a variables plus at most b others. This notion of robustness implicitly assumes that there is a limited amount of time available to re-schedule if the scenario which was planned for changes. Then, super solutions have the additional benefit that the original plan only needs to be adapted slightly to be legal. A modification of this idea is presented in [6] where the notion of multi-consistency is introduced. The idea of computing a set of back-up solutions is taken to extremes in [10] where linear programming sensitivity analysis is used to compute a potentially exponentially large set of optimal solutions for linear programs with changing objectives.

Contribution: In this paper, we assume that there is extensive time available before the day of operation when our solution will need to be realized, but that during this period the problem is associated with uncertainty. Then, when the exact problem is revealed to us, there is only very little time left to compute a high-quality feasible solution. Under which conditions, and how, can we efficiently prepare ourselves in that situation? In more formal terms, we address the question whether we can trade time for uncertainty by precomputing a set of solutions (our "plan Bs") such that, for every possible scenario, there exists a feasible solution in our set that is near optimal. Applied to the example of the delivery company, we investigate whether a set of optimization problems could be solved overnight so that we are guaranteed to have a near-optimal schedule waiting to be executed for the scenario that we are facing in the morning.

We do not assume that stochastic information is available. Like earlier work on uncertain constraint satisfaction problems (see, for example, [15] and [16]), we consider linear and integer linear programming (LP and IP) formulations. LPs and IPs are known to be capable of modelling a large variety of combinatorial optimization problems, and can be used efficiently in problems such as AI planning [14]. However, we focus on LPs and IPs for a different reason, namely because there exists an established standard form that, as we shall see, we can exploit both to define uncertainties as well as to determine what scenarios should be considered to prepare ourselves effectively for the future.

We make two important remarks: First, the uncertainties that we consider are defined in terms of linear programming formulations while, a priori, it is unclear how uncertainties in problem parameters translate into uncertain parameters of a linear programming formulation. Note that, in principle, it is possible that one uncertainty translates into many correlated uncertain parameters in the corresponding linear programming model. Extra care must be taken to decide in which scenarios the following work applies. Second, we use (integer) linear programs strictly as modelling entities, while the solution method for the problems that we consider can always be chosen freely.

The class of uncertain linear programs that we consider represents a subset of so-called linear programs with interval coefficients (LPICs). LPICs were introduced in [5] where an exponential-time algorithm was presented that allows to assess the range in which the optimal objective function value can lie depending on what values the coefficients in their respective interval will take. This gap-analysis is not trivial because LPICs allow unsigned variables and equality constraints. Excluding these two cases, we develop a method that provides worst case quality guarantees for uncertain LPs and IPs. We show that, even with signed variables and in the absence of equality constraints, no polynomial number of precomputed solutions suffices to prepare for all possible future scenarios. Despite this negative result, for a variety of more restricted cases, we can devise polynomial time algorithms that allow to find a near-optimal feasible solution in sublinear time as soon as all uncertainty has vanished. The restrictions guarantee, in particular, that infeasibility will not occur.

2 Uncertain Linear Programs

Definition 1 **(ϵ-approximation).** *Consider the problem $L = (P, f)$ of maximizing a function f over a set P into the non-negative rationals, and let OPT(L) denote the optimal value $\max_P f$, if it exists. Given $\epsilon > 0$, an* ϵ-approximation *is an element $x \in P$ such that $(1+\epsilon)f(x) \geq f(y)$ for all $y \in P$.*

We use the following paradigm. Given partial knowledge on the optimization problem, precompute a set of candidate solutions such that, once, at a later time, (P, f) is known exactly, then it is possible to use that prearrangement so as to find very fast (in sublinear time) a near-optimal solution to the problem. In this section we show how and under what circumstances this can be achieved for LPs.

For any two matrices or vectors, we write $A \geq B$ when this inequality holds for every component. Henceforth we assume that we are given two m by n matrices $\underline{A} \leq \overline{A}$, two vectors $\underline{b} \leq \overline{b} \in \mathbb{Q}^m$, two vectors $\underline{p} \leq \overline{p} \in \mathbb{Q}^n$, a desired approximation guarantee $\epsilon > 0$, and a constant $k \in \mathbb{N}$.

Definition 2 **(Uncertain Linear Program and Prearrangement).** *An* uncertain linear program (ULP) *S_{LP} is given as $(\underline{A}, \overline{A}, \underline{b}, \overline{b}, \underline{p}, \overline{p})$ and consists of the set of all LPs of the form*

$$\max p \cdot x$$
$$\text{s.t.} \begin{cases} Ax \leq b \\ x \quad \geq 0, \end{cases}$$

such that $\underline{A} \leq A \leq \overline{A}$, $\underline{b} \leq b \leq \overline{b}$, and $\underline{p} \leq p \leq \overline{p}$.

A prearrangement *of S_{LP} with parameter ϵ is a set X of rational vectors such that for every linear program $L = (A, b, p)$ in S_{LP}, there exists an ϵ-approximation in X.*

ULPs, as defined above, represent a subset of linear programs with interval coefficients (LPICs) that were introduced in [5]. Note that, in ULPs, equality constraints are not allowed, and neither are unsigned variables. In order to avoid infeasibility issues we further restrict the ranges to positive intervals:

Definition 3 **(Range-positive).** *Given a ULP, we say that it is* range-positive wrt *A, iff (1) all coefficients of $\underline{A}$ are non-negative: $\underline{A} \geq 0$, and moreover, (2) uncertain coefficients must be positive: $\overline{a}_{ij} \neq \underline{a}_{ij} \Rightarrow \underline{a}_{ij} > 0$. Similarly, we say that it is range-positive wrt b (resp. p) iff (1) all coefficients of $\underline{b}$ (resp. $\underline{p}$) are non-negative, and moreover, (2) uncertain coefficients must be positive.*

We will also need to restrict the number of parameters that can vary more than marginally. We introduce the following notation:

Given two vectors (or matrices) v, w, and given $\delta \geq 0$ we write $v \geq_k^\delta w$, iff $v \geq w$ and for all but k coordinates, it holds that $w_i \leq v_i \leq w_i + \delta|w_i|$.

Definition 4 **(Bounded Uncertainties).** *Given a ULP S_{LP}, $k \in \mathbb{N}$, and $\delta > 0$, a* Linear Program with k-δ-Bounded Uncertainties *is the subset $S_{BLP} \subset S_{LP}$ consisting of the LPs of S_{LP} such that $A \geq_k^\delta \underline{A}$, $b \geq_k^\delta \underline{b}$ and $p \geq_k^\delta \underline{p}$.*

To ease the presentation, we use the lower end of each interval as the default value of the coefficient, but this is by no means necessary: the default value for each coefficient could also be explicitly specified and lie anywhere in the given range.

Note that Definition 4 does not require that we know *which* k parameters in A, b, and p will change by more than a small factor from their default value. We will discuss in a moment how knowledge about which k parameters could change more than marginally can be exploited to yield to more efficient prearrangement algorithms.

When starting to think about the problem, at first, even when only one parameter in an LP can change over a larger range, it is not clear that there exists a polynomial-size prearrangement. For this, we need to prove that, when a range-positive parameter in the LP is changed slightly, then the objective function can only change marginally, too. From this, it follows that, for range-positive ULPs with bounded uncertainties, we can efficiently compute a prearrangement that will allow us to lookup a solution later in sublinear time.

Theorem 1. *Given $\epsilon > 0$, let S_{BLP} denote a ULP with k-$\epsilon/4$-bounded uncertainties that is range-positive wrt A, b, and p. Let C denote the maximum of all ratios $\overline{p}_i/\underline{p}_i$, $\overline{b}_j/\underline{b}_j$, and $\overline{a}_{ij}/\underline{a}_{ij}$ (where by convention $0/0 = 1$). We can compute a prearrangement X of size polynomial in m, n, $\log C$, and $1/\epsilon$. The computation time of X equals $|X|$ times the time to solve an LP of S_{BLP}.*

Moreover, once X is computed, given the coefficients of the LP which deviate more than marginally from their anticipated value, one can compute an ϵ-approximation of the modified problem P in time sublinear in the size of the ULP.

Proof: Here is the algorithm.

Prearrangement: Let $\delta := \epsilon/4$. For each subset I_p of $[1, n]$ of size k, each subset I_b of $[1, m]$ of size k, and each subset I_A of $[1, m] \times [1, n]$ of size k, let $n_p^i = \log_{1+\delta}(\overline{p}_i/\underline{p}_i)$ for $i \in I_p$, $n_b^i = \log_{1+\delta}(\overline{b}_i/\underline{b}_i)$ for $i \in I_b$, and $n_A^{ij} = \log_{1+\delta}(\overline{a}_{ij}/\underline{a}_{ij})$ for $(i, j) \in I_A$.

For each list of integers $((k_i)_{i\in I_p}, (\ell_i)_{i\in I_b}, (m_{ij})_{(i,j)\in I_A})$ such that $0 \le k_i \le n_p^i$, $0 \le \ell_i \le n_b^i$, and $0 \le m_{ij} \le n_A^{ij}$, consider the LP defined by

$$p_i = \begin{cases} \underline{p}_i & \text{if } i \notin I_p \\ \underline{p}_i(1+\delta)^{k_i} & \text{if } i \in I_p \end{cases},$$

$$b_i = \begin{cases} \underline{b}_i & \text{if } i \notin I_b \\ \underline{b}_i(1+\delta)^{\ell_i} & \text{if } i \in I_b \end{cases}, \text{ and}$$

$$a_{ij} = \begin{cases} \underline{a}_{ij} & \text{if } \overline{a}_{ij} = \underline{a}_{ij}, \text{ else} \\ \underline{a}_{ij}(1+\delta) & \text{if } (i,j) \notin I_A \\ \underline{a}_{ij}(1+\delta)^{m_{ij}} & \text{if } i \in I_A \end{cases}.$$

Notice that, since all coefficients a_{ij} and b_i are non-negative, 0 is always a feasible solution of this LP, hence there always is a (possibly infinite) optimal solution. Compute an optimal solution, and add the LP and its solution to X. We store the elements of X represented as $((k_i)_{i\in I_p}, (\ell_i)_{i\in I_b}, (m_{ij})_{(i,j)\in I_A})$ in a data structure enabling fast search, say, a balanced tree for example (whereby we use the lexicographic order on the representing tuples to define an order on the LPs).

Solution Lookup: Given X, we can gather an ϵ-approximation for any given LP $P = (A, b, p) \in S_{BLP}$ in sublinear time. When we are given a list of at most k coefficients and their new values, we compute $((k_i)_{i \in I_p}, (\ell_i)_{i \in I_b}, (m_{ij})_{(i,j) \in I_A})$ such that:

- $\forall\, i \in I_p :\ \underline{p}_i(1+\delta)^{k_i - 1} < p_i \leq \underline{p}_i(1+\delta)^{k_i}$.
- $\forall\, i \in I_b :\ \underline{b}_i(1+\delta)^{\ell_i} \leq b_i < \underline{b}_i(1+\delta)^{\ell_i + 1}$.
- $\forall\, (i,j) \in I_A :\ \underline{a}_{ij}(1+\delta)^{m_{ij} - 1} < a_{ij} \leq \underline{a}_{ij}(1+\delta)^{m_{ij}}$.

Then, we search and output the optimal solution of the corresponding LP P' in X.

Correctness: We first claim that any feasible solution x' of P' is also a feasible solution of program P. Indeed, by definition $x' \geq 0$ and $A'x' \leq b'$. Thus:

$$Ax' \leq A'x' \leq b' \leq b. \tag{1}$$

Second, let us analyze the quality of the solution that we return. Let x denote an optimal solution for P. We claim that $x/(1+\delta)^2$ is a feasible solution for P':

$$A'x/(1+\delta)^2 \leq Ax/(1+\delta) \leq b/(1+\delta) \leq b' \tag{2}$$

and the scaled solution remains non-negative. Since x' is an optimal solution for P', $p' \geq p$, and $\delta = \epsilon/4$, we have

$$p \cdot x' \geq p' \cdot x'/(1+\delta) \geq p' \cdot x/(1+\delta)^3 \geq p \cdot x/(1+\epsilon). \tag{3}$$

Complexity: The set X has size at most $4\binom{n}{k}\binom{m}{k}\binom{mn}{k}\log^{3k}(C)/\epsilon$. Note that this is polynomial in the description size of the instance of the problem for constant k. Constructing X amounts to solving $|X|$ LPs from S_{BLP}, which can thus be done in polynomial time.

Regarding the lookup of a solution, we see that the computation of the representing tuple can then be performed in time $O(3k \log\log C)$. Finding the corresponding solution then can be done in time $O(\log |X|)$. Since $|X|$ is polynomial in the size of the given ULP and $1/\epsilon$, the total lookup time is in $O(\log(1/\epsilon) + \log(|S_{BLP}|) + k)$ where $|S_{BLP}|$ denotes the input size of the ULP. Consequently, the lookup can be performed in time sublinear in the size of the ULP. $\square$

Note that Theorem 1 considers the most general case where all parameters can change, k parameters can change more than just marginally, and we have absolutely no idea which ones. If we knew those k parameters in advance, the theorem implies that a prearrangement of size $O(\log^{3k}(C)/\epsilon)$ is sufficient, which is very manageable and quite encouraging. Between the two extremes, whenever the size of the set from which the k non-marginal parameters are drawn from can be limited, this knowledge can be leveraged to make much smaller prearrangements.

We now prove that the assumption of bounded uncertainties of Theorem 1 is not arbitrary but necessary.

Theorem 2. *There exists a ULP S_{LP} which is range-positive wrt A, b, and p, and such that every prearrangement X for S_{LP} has size at least exponential in the size of the description of S_{LP}.*

Proof: Consider the following ULP, where $m = n$. $\underline{A} = \overline{A}$ is the identity matrix, $\underline{p} = \overline{p}$ is the all-ones vector, $\underline{b}$ is the all-ones vector, and $\overline{b}$ is the vector whose entries are all equal to 2. In other words, S_{LP} is the set of LPs of the form:

$$\max \sum_i x_i$$

$$s.t. \begin{cases} \forall i, & x_i \leq b_i \\ \forall i, & x_i \geq 0, \end{cases}$$

where for each i we have $1 \leq b_i \leq 2$.

Any prearrangement of S_{LP} must also be a prearrangement of the subset S' consisting of those LPs such that exactly half of the b_i's are equal to 1 and the other half are equal to 2. Any such LP has value $(n/2) + 2(n/2) = 3n/2$. Let $x \in X$ be in a prearrangement of S'. Without loss of generality, we can assume that every coordinate x_i is either 1 or 2. Setting $\xi := \epsilon/(1+\epsilon)$, we must have: $\sum_i x_i \geq (1-\xi)3n/2$, so the number k of ones in x satisfies

$$k + 2(n-k) \geq (1-\xi)3n/2,$$

hence $k \leq n/2 + 3\xi n/2$. But x can only be a feasible solution for at most $\binom{k}{k-n/2}$ elements of S', which is at most $\binom{n/2+3\xi n/2}{3\xi n/2}$. Since S' has cardinality $\binom{n}{n/2}$, any prearrangement of S' must have cardinality at least

$$\frac{\binom{n}{n/2}}{\binom{n/2+3\xi n/2}{3\xi n/2}},$$

which is exponential in n for $\xi < 1/3$ or $\epsilon < 1/2$. By scaling the right hand side, we can generalize the result for all $\xi < 1$ and $\epsilon > 0$. □

At this stage, the significance of our positive result is rather limited since we can really only handle LPs with non-negative coefficients and $\leq$ constraints. We find that analogous results hold for LPs of the form: $\min p \cdot x$ such that $Ax \geq b$, $x \geq 0$ (in other words, we can tackle both "covering" and "packing" linear programs). However, one may wonder how necessary is it to assume that the coefficients are all non-negative. We now prove that the range-positive assumptions of Theorem 1 can be somewhat relaxed, although at the cost of restricting the uncertainty.

Theorem 3. *Let S_{BLP} denote a ULP with bounded uncertainties. Assume that at least one of the following sets of conditions hold.*

- $\underline{A} = \overline{A}$ *and S_{BLP} is range-positive wrt b and p.*
- $\underline{b} = \overline{b}$ *and S_{BLP} is range-positive wrt A and p.*
- $\underline{b} = \overline{b}$, $\underline{p} = \overline{p}$, *and S_{BLP} is range-positive wrt A.*
- $\underline{A} = \overline{A}$, $\underline{b} = \overline{b}$, *and S_{BLP} is range-positive wrt p.*

Then the conclusion of Theorem 1 still holds.

Proof: The algorithm is identical to the one used in the proof of Theorem 1 except for a slightly improved setting of δ. Then, only minor differences occur in the analysis of correctness.

- Under the first set of conditions, we set $\delta := \epsilon/3$. In the analysis, we then replace Equation 2 by:

$$A'x/(1+\delta) = Ax/(1+\delta) \leq b/(1+\delta) \leq b',$$

  thus proving that $x/(1+\delta)$ is feasible for P'. We can then replace Equation 3 by:

$$p \cdot x' \geq p' \cdot x'/(1+\delta) \geq p' \cdot x/(1+\delta)^2 \geq p \cdot x/(1+\epsilon).$$

  Notice that it may now happen that P is infeasible, but that this is detected by the algorithm, since P is infeasible iff P' is also infeasible.
- Under the second set of conditions, either $\underline{b} \geq 0$ in which case the problem is range-positive wrt b and we can apply Theorem 1, or there exists no feasible solution to any LP in S_{BLP}.
- Under the third set of conditions, we set $\delta := \epsilon$ and replace Equation 2 by:

$$A'x/(1+\delta) \leq Ax \leq b = b',$$

  thus proving that $x/(1+\delta)$ is feasible for P'. We can then replace Equation 3 by:

$$p \cdot x' = p' \cdot x' \geq p' \cdot x/(1+\delta) = p \cdot x/(1+\delta) = p \cdot x/(1+\epsilon).$$

- Under the last set of conditions, we set $\delta := \epsilon$. We notice that x is trivially feasible for P'. Therefore, we can replace Equation 3 by:

$$p \cdot x' \geq p' \cdot x'/(1+\delta) \geq p' \cdot x/(1+\delta) \geq p \cdot x/(1+\epsilon).$$

□

By slight adaptation of our algorithm and its analysis, it is easy to show that analogous results still hold when we consider ULPs of the form $\min p \cdot x$ such that $Ax \geq b$ and $x \geq 0$. As a consequence, by arguing about linear programming duality, we can further extend Theorem 1 as follows.

Corollary 1. *The conclusion of Theorem 1 is still valid when one of the following holds:*

- $\underline{p} = \overline{p}$ *and* S_{BLP} *is range-positive wrt* A *and* b*.*
- $\underline{A} = \overline{A}$*,* $\underline{p} = \overline{p}$*, and* S_{BLP} *is range-positive wrt* b*.*

What we have achieved now is that we can handle arbitrary ULPs with bounded uncertainties where only the part (constraint matrix, objective or right hand side) that is subject to potential changes is required to be range-positive. We believe that this requirement is mild enough to allow to tackle practical applications where it is often the case that objective and/or right hand side entries all have the same sign.

3 Prearranging Integer Programs

For LPs, we studied under which conditions prearrangement can be performed efficiently. We envision applications of this work for example in very time critical, real-time systems like sensor networks where changing volumes of data need to be managed very quickly which does not allow to re-solve larger LPs. However, one may argue that, since LPs can be solved in polynomial time, the number of applications is limited in which reducing the complexity to sublinear time is really necessary. Therefore, in this section we study integer linear programs that are NP-hard to solve in general.

Definition 5 (Uncertain Integer Programs). *The* uncertain integer program (UIP) S_{IP} *consists of the set of all IPs of the form*

$$\max p \cdot x$$

$$\textit{s.t.} \begin{cases} Ax \leq b \\ x \quad \geq 0 \\ x \quad \textit{integer}, \end{cases}$$

such that $\underline{A} \leq A \leq \overline{A}$, $\underline{b} \leq b \leq \overline{b}$ *and* $\underline{p} \leq p \leq \overline{p}$.

A prearrangement *of S_{IP} with parameter ϵ is a set X of integer vectors such that for every IP $I = (A, b, p)$ in S_{IP}, there exists an $x \in X$ which is feasible for I and satisfies $(1+\epsilon)p \cdot x \geq OPT(I)$.*

Range-positiveness and IPs with bounded uncertainties (UIPs) can be defined analogue to Definition 3 and Definition 4. First, we show that the conditions that need to hold to allow for an efficient prearrangement for UIPs are strictly stronger than for ULPs:

Theorem 4. *There exist UIPs which cannot be prearranged efficiently, but whose linear relaxations can.*

Proof: For all $M \in \mathbb{N}$, $M > 1$, consider the following UIP:

$$\max x_1 + 2\epsilon x_2$$

$$\text{s.t.} \begin{cases} x_3 & \leq b_1 \\ Mx_2 - x_3 & \leq b_2 \\ x_1, x_2 & \leq 1 \\ x_1, x_2, x_3 & \geq 0 \\ x_1, x_2, x_3 & \text{integer}, \end{cases}$$

where $b_1, b_2 \in [1, \ldots, M-1]$.

Since only two components can change, the UIPs above have 2-0-bounded uncertainties, of course. Note also, that the constraint matrix A and the objective $\underline{p}$ (with respect to which the UIP is range-positive) are not allowed to change at all and that the UIP is range-positive wrt b. Consequently, according to Theorem 3, its corresponding linear relaxation ULP can be prearranged efficiently.

Now, for all b_1, b_2 such that $b_1 + b_2 < M$, the optimal objective value is 1. As soon as $b_1 + b_2 = M$ the objective suddenly raises to $1 + 2\epsilon$. The optimal solutions for all the $M - 1$ scenarios where $b_1 + b_2 = M$ are all different from each other and no one solution that is feasible for 2 of them achieves an objective value of more than 1. Consequently, a feasible prearrangement for this class of UIPs must contain at least M solutions, which is exponential in the size of the input. $\square$

The intractability in the above example is solely caused by infeasibility of solutions that could otherwise represent several scenarios. We achieve our first positive result for UIPs for cases where feasibility is not an issue.

Theorem 5. *Assume we are given $k \in \mathbb{N}$, $\epsilon > 0$ and $S_{BIP} = (\underline{A}, \overline{A}, \underline{b}, \overline{b}, \underline{p}, \overline{p})$ with k-ϵ-bounded uncertainties that is range-positive wrt p and for which $\underline{A} = \overline{A}$ and $\underline{b} = \overline{b}$. Then S_{BIP} can be prearranged efficiently in the sense of Theorem 1.*

Proof: We use the same algorithm as in Theorem 1 with the modification that we set $\delta := \epsilon$. Since $\underline{A} = \overline{A}$ and $\underline{b} = \overline{b}$, we have that $I_A = \emptyset$ and $I_b = \emptyset$. Therefore, for each element $(k_i)_{i \in I_p}$, we consider the IP defined by

$$p_i = \begin{cases} \underline{p}_i & \text{if } i \notin I_p \\ \underline{p}_i (1+\delta)^{k_i} & \text{if } i \in I_p \end{cases},$$

$b_i = \underline{b}_i$ and $a_{ij} = \underline{a}_{ij}$. The lookup of x' works in the same way as for ULPs. Since constraint matrix and right hand side are constant, feasibility of the returned solution is not an issue in this case. Regarding the quality of the returned solution, let us denote the optimal solution for P by x. Then, the same inequality as in the last case of Theorem 3 proves the desired accuracy.

Finally, we analyze the efficiency of this algorithm. The set X has size at most $\binom{n}{k} \log^k(C)/\epsilon$. Constructing X amounts to solving $|X|$ LPs from S_{BIP}. The computation of the representing tuple can then be performed in time $O(k \log\log C)$. Then, the total lookup time is in $O(\log(1/\epsilon) + \log(|S_{BIP}|) + k)$ where $|S_{BIP}|$ denotes the input size of the UIP. Consequently, the lookup can be performed in time sublinear in the size of the input after a polynomial number of IPs have been solved in the prearrangement phase. $\square$

Again, note that the size of the prearrangement decreases to $O(\log^k(C)/\epsilon)$ when we know which k parameters in the objective can change more than marginally. We have seen before that even changing only two components in the constraint structure of the problem can yield to intractable prearrangement problems. The interesting question arises whether we can afford any change within A or b at all. The following result answers this question affirmatively.

Theorem 6. *Let S_{BIP} be a UIP with all integer coefficients in the matrix and the right hand side. Assume also that exactly one matrix or right hand side coefficient c can vary: $\underline{c} \le c \le \overline{c}$, and such that the optimal solution for the most restricted problem in S_{BIP} has a strictly positive value $\underline{z} > 0$, while the optimal solution to the least restricted problem has value $\overline{z} < \infty$. Then we can compute efficiently a prearrangement X of size $|X| \in O(mn(\log(\overline{z}/\underline{z})/\log(1+\epsilon)))$.*

Proof: The prearrangement algorithm that we propose is quite simple: We try to assemble a sequence of scenarios P_i where P_0 solves the most restricted problem (i.e. when c is a coefficient in b we consider the problem with $c = \underline{c}$, otherwise the one where $c = \overline{c}$). Then, for $i > 0$ we want P_i to be the problem where $\underline{c} \leq c \leq \overline{c}$ is minimal (maximal) and $P_i \geq (1+\epsilon)P_{i-1}$, where c is a coefficient in b (A). We iterate this process until no subsequent P_i can be found anymore. Clearly, the number of problems that we assemble that way is bounded by $mn(\log(\overline{z}/\underline{z})/\log(1+\epsilon))$. Also, this set of solutions is a feasible prearrangement in that an ϵ-approximation for every possible scenario exists in our set. The question remains how we can find the subsequent P_i's efficiently. This is where we exploit the fact that only one coefficient can change: We simply perform a binary search for the critical value of c where the next level in the objective is reached. The total number of problems that need to be solved is then in $O(mn(\log(\overline{z}/\underline{z})\log(\overline{c}-\underline{c})/\log(1+\epsilon)))$, which is polynomial in the size of the input. □

Interestingly, the search for a critical coefficient in b or A where a jump in the objective occurs can be combined with the result that we achieved Theorem 5: for each problem considered in the algorithm for UIPs with changing objectives we can perform a binary search on the one changing component in A or b and therefore prearrange UIPs with bounded uncertainties where at least all but one unknown coefficient are located in the range-positive objective function.

The last result that we report is probably the most important of all, even though it is really a simple implication of our results on ULPs. When looking back at the example where the change of two coefficients yielded an intractable integer prearrangement problem, we find that the gap in the objective value between integer solution and the best solution of the linear relaxation of the problem is quite large. It is straightforward that we can approximate the accuracy defined by that LP-IP gap by using the algorithm from Theorem 1. Formally, we achieve:

Theorem 7. *Assume we are given $k \in \mathbb{N}$, $\epsilon > 0$, and a UIP S_{BIP} with k-$\epsilon/4$-bounded uncertainties where each part (A, b, or p) where potential changes can occur is range-positive. If $g > 0$ is a lower bound on the minimal ratio between an IP in S_{BIP} and its linear continuous relaxation, then we can efficiently compute a prearrangement that achieves an approximation guarantee of $(1+\epsilon)/g$.*

4 Practical Considerations

In this final section, we would like to make a few comments on the practicability of the algorithms presented here. While at the end of the discussion on ULPs we found that, if solving the problem is too easy, in many practical applications it may not be necessary at all to plan ahead. After reading our last section that tackles UIPs, one may now argue that solving a polynomial number of NP-hard problems in the prearrangement phase may be asked too much. Just like humans usually only plan ahead in this manner when a lot is at stake, we will probably only want to go through this hassle when having a near optimal solution in some uncertain environment is critical. Or, for example, when we design a system that needs to work well in changing environments and where the time that is spent to "prepare" the system (for example before the software is launched) is much less important than a good performance later.

While the work presented here is theoretical in nature, we would like to note that what we really did was to analyze and adapt a strategy that is used in practice not only by human beings, but also by existing computer systems, for instance for parametric query optimization in distributed data base systems as cited in [10]. Therefore, prearrangement must be viewed as a method that is used in practice — but to the best of our knowledge it has not been analyzed theoretically before. Our analysis revealed some sufficient conditions under which efficient prearrangement is possible at all. Moreover, we give a strategy how the scenarios for which one should plan ahead ought to be selected.

We discussed already how knowledge about the concrete non-marginal, uncertain parameters can be leveraged to decrease the size of a prearrangement significantly. Note that, in practice, we can further reduce the computational effort needed by compromising on the solution quality. By reversing the arguments in this paper, if a limit on the maximum size of the prearrangement is given, our method also allows to pick scenarios so that the worst-case error made is minimized. For instance, in the introductory example of the delivery company, we may find that there is time to compute 120 problems on the four computers in their office. Then, our method can be used to decide for which scenarios we should prepare a schedule. Moreover, when stochastic information is available which and how coefficients in the problem can differ, then by sampling over the parameters $((k_i)_{i\in I_p}, (\ell_i)_{i\in I_b}, (m_{ij})_{(i,j)\in I_A})$ we can precompute a set of solutions that maximizes the likelihood that an ϵ-approximate solution will be available.

For UIPs, another possible scenario is that an approximation algorithm for the problem at hand is available. Then we can precompute approximate solutions rather than solving each IP to optimality. Our prearrangement algorithm will then approximate with the performance guarantee of the approximation algorithm that is being used plus the additional error caused by the LP-IP gap. Even more, if the analysis of the approximation algorithm itself is based on the LP-IP gap then we even achieve that same approximation guarantee through prearrangement, even if only approximate solutions have been precomputed!

Furthermore, our method is naturally parallelizable and it also works when the LP-IP gap is not known. That way, if we have a complex problem to solve for which we believe that the LP provides a good upper bound, then we have a method at hand that prepares us for the future in a very reasonable manner.

As a final remark, it is a matter of future work to determine whether linear programming sensitivity analysis could allow us to significantly reduce the work that needs to be conducted for prearrangement. In practice one would expect to reduce the computational effort considerably by realizing that many parts of an LP can change significantly before affecting the optimal solution at all.[1] It is questionable, however, whether this important practical improvement could actually result in better worst case guarantees like the ones that were the target of this study.

[1] We owe this observation to an anonymous referee.

5 Conclusions

We investigated one of the most common ways how humans cope with uncertainty, namely by the provision of backup plans. While this approach is frequently being used in time-critical computer systems, to the best of our knowledge no theoretical analysis has been provided before. Our assumption is that there exists an extensive time period before the day of operation where the problem is still associated with uncertainty. Then, when the real problem that needs to be accommodated is revealed to us, there is very little time available before a solution needs to realized.

Within this scenario, we studied uncertain linear and integer linear programs. We showed that bounding the number of uncertain coefficients is essential to allow for an efficient prearrangement, whereby we do not necessarily need to know exactly which coefficients will change more than marginally while this knowledge can be used to reduce the computational efforts. For uncertain linear programs, we specified a method for choosing a polynomial number of scenarios whose pre-solution guarantees that an ϵ-approximation of the real problem will be available. For this method, it is essential that every part of the LP (constraint matrix, right hand side, or objective) that is subject to potential changes is range-positive. For uncertain integer programs, we found analogous results as for LPs when only the range-positive objective function is subject to changes or when the desired approximation guarantee is not larger than the LP-IP gap. Especially when it is known in advance which uncertain parameters can change more than just marginally, our method yields to prearrangements of very manageable size.

There are some open questions: It can be shown that our method breaks down when value ranges include 0 or when both negative and positive ranges are mixed within the same part of the LP. Is there another way of choosing scenarios that can accommodate these cases? Are there other conditions under which an efficient prearrangement of uncertain integer programs is possible?

References

1. I. Althoefer, F. Berger, S. Schwarz. Generating True Alternatives with a Penalty Method. http://www.minet.uni-jena.de/MathNet/reports/shadows/-02-04report.html, 2002.
2. A. Ben-Tal, A. Nemirovski. Robust Optimization - Methodology and Applications. *Mathematical Programming Series B*, 92:453–480, 2002.
3. J. Birge, F. Louveaux. Introduction to Stochastic Programming. *Springer*, 1997.
4. A. Borodin, R. El-Yaniv. Online Computation and Competitive Analysis. *Cambridge University Press*, 1998.
5. J.W. Chinneck and K. Ramadan. Linear programming with interval coefficients. *Journal of the Operational Research Society*, 51(2):209–220, 2000.
6. K. Elbassioni, I. Katriel. Multiconsistency and Robustness with Global Constraints. *CP-AI-OR*, 2005.
7. D. Eppstein. Bibliography on k shortest paths and other "k best solutions" problems. http://www.ics.uci.edu/∼eppstein/bibs/kpath.bib, 2001.
8. C. Guestrin, D. Koller, C. Gearhart, N. Kanodia. Generalizing Plans to New Environments in Relational MDPs. *IJCAI*, 2003.
9. E. Hebrard, B. Hnich, T. Walsh. Super Solutions in Constraint Programming. *CP-AI-OR*, LNCS 3011:157–172, 2004.
10. A. Hulgeri, S. Sudarshan. Parametric Query Optimization for Linear and Piecewise Linear Cost Functions. *VLDB*, 167–178, 2002.
11. M. Lagoudakis, R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.
12. I. Sameith. On the Generation of Alternative Solutions for Discrete Optimization Problems with Uncertain Data. http://www.minet.uni-jena.de/MathNet/reports/-shadows/04-01report.html, 2004.
13. B. Verweij, S. Ahmed, A. Kleywegt, G. Nemhauser, A. Shapiro. The Sample Average Approximation Method Applied to Stochastic Routing Problems: A Computational Study. *Computational Optimization and Applications*, 24(2-3):289–333, 2003.
14. T. Vossen, M. Ball, A. Lotem, D.S. Nau. On the Use of Integer Programming Models in AI Planning. *IJCAI*, 1999.
15. N. Yorke-Smith. Reliable Constraint Reasoning with Uncertain Data. PhD thesis, IC-Parc, Imperial College London, June 2004.
16. N. Yorke-Smith and C. Gervet. Tight and Tractable Reformulations for Uncertain CSPs. Proceedings of CP'04 Workshop on Modelling and Reformulating Constraint Satisfaction Problems, Toronto, Canada, September 2004.