Algorithms and Concepts for Robust Optimization

Dissertation zur Erlangung
des mathematisch-naturwissenschaftlichen Doktorgrades
Doctor rerum naturalium
der Georg-August Universität Göttingen

vorgelegt von
Marc Goerigk
aus Berlin

Göttingen 2012

Referentin:	Prof. Dr. Anita Schöbel
Koreferent:	Prof. Dr. Marco Lübbecke
Tag der mündlichen Prüfung	24.9.2012

Preface

The more I occupied myself with robust optimization, the more it grew from a mathematical discipline to a pattern of thought that influenced my everyday decisions. Uncertainty is all around us - from the flight to Tromsø or the train to Saarbrücken, to the details like lactose in sandwiches (better don't try if you're not sure).

Along my way I had plenty of people who helped me and generously overlooked some overconservatism stemming from restrictive robustness concepts.

First of all, I thank Anita Schöbel for endless support and her impressive ability to not sleep - at least, that's how it seems. Also the great spirit of the optimization working group at the Institute for Numerical and Applied Mathematics made writing this thesis a joy: Thanks to Marco Bender, Ruth Hübner, Jonas Ide ("your partner in science"), Mark Körner ("der große Mark"), Thorsten Krempasky ("Dr. Lightning"), Michael Schachtebeck, Robert Schieweck, Marie Schmidt ("der kleine Marc"), Daniel Scholz, Morten Tiedemann and Stephan Westphal. Also, the proof-reading of Marie, Ruth, Marco and Robert was a great help during the final phase of the work.

I thank Florian Bruns, Emilio Carrizosa, Markus Chimani, Martin Knoth, Sigrid Knust, Matthias Müller-Hannemann, Petra Mutzel, and Bernd Zey for the inspirational and challenging joint work (see Chapter 8 for an overview). During my time in New Zealand, I enjoyed the hospitality of Matthias Ehrgott, Andrea Raith and Fritz Raffensperger, who made me feel at home when I was in fact up to an $\varepsilon > 0$ at maximum distance. Furthermore, Carsten Damm and Max Wardetzky were the best tutor team I could have had.

Finally, I thank Christiane for her endless patience and support.

Contents

If a man will begin with certainties, he shall end in doubts;
but if he will be content to begin with doubts, he shall end in certainties.
-Francis Bacon, The Advancement of Learning

1 Introduction

In this Chapter we consider the foundations of this work: After a short motivation on the use of robust optimization, we formally introduce uncertain optimization problems, and lay out the structure, methodology and purpose of this work.

1.1 Motivation

In this thesis we deal with uncertainty – and how to end in certainty. It is a topic everybody has collected their own experience with.

Let's assume you have a lot of appointments and activities scheduled for today, and let's further assume that reading into this work is one of them – one of the more pleasurable ones, of course! As you are likely to be a mathematically inclined person, you might have put some thought into optimizing today's schedule, assuming given durations for each appointment.

However, what if this work catches you so compellingly that you find the originally scheduled time horizon insufficient? Or, on the other extreme, what if you find it so annoying you will barely read more than this very sentence?

Even though the latter does not seem to be the case, you might come to the conclusion that your original schedule does not suit your needs anymore, and you will change it accordingly. Robust optimization is the mathematical discipline that takes exactly this uncertainty in the problem parameters into account – by finding solutions that are still "good" when things happen to turn out differently.

How exactly this is done depends on the *robustness concept* applied. Depending on your situation and character, you might prefer solutions that suit all possible parameter realizations, solutions that can be cheaply repaired, or solutions that distribute a given buffer budget over the critical times of the day.

Robust optimization is not new – in fact, its beginnings trace back to the 1970s. However, partly due to the usually increased problem size when optimizing under uncertainty, there has been a surge of literature only since the late nineties, when increased computational power and software sophistication allowed to handle such complex models.

Since then, a large pool of robustness concepts have evolved, each of them with their own advantages and disadvantages. As it is said in [Sti08]: "There is no silver bullet for optimization under imperfect information.", and no concept fits all needs.

Grossly oversimplifying the current situation of research in robust optimization, there are basically two types of approaches: More theoretically-driven ones, that tend to produce models that can hardly be used for real-world problems, and application-driven ones, that are often so narrowly tailored, they can barely be transferred to other applications. In this work we try to bring the field of robust optimization further, in an effort to bring theory and practice one step closer together.

1.2 Uncertain Optimization

We now introduce the general framework of uncertain problems that form the starting point for robust optimization.

Uncertainty. Nearly every optimization problem suffers from uncertainty to some degree, even if this does not seem to be the case at first sight. Two types of uncertainty can be distinguished: *Microscopic* and *macroscopic* uncertainty.

Microscopic uncertainty includes:

1. **Numerical errors.** Storing any number on a computer system is only possible up to a certain exactness, resulting in so-called floating-point errors that may propagate. Although exact mixed-integer programming solvers do exist (e.g., [CKSW11]), they are typically several orders of magnitude slower than competitive floating-point optimization software, and therefore not industrial standard. In a representative experiment using NetLib instances, [BTN00] report that optimal solutions to 13 of 90 considered instances violate constraints by more than 50%, when "ugly" coefficients are perturbed by only 0.01%. As these coefficients are most likely not exact, but only truncated values, this is a warning that "small" inaccuracy can create "very bad" solutions.

2. **Measurement errors.** Whenever mathematical models are applied to real-world problems, they need to be supplied with data that was measured in some way. These measurements may be intrinsically inexact, e.g., when the volume or the weight of a body is determined, or only statistically representative, e.g., when stemming from a survey. Even though these values may seem "reasonably exact", we see from the previous paragraph that their impact on the solution usefulness can be large.

Macroscopic uncertainty includes:

3. **Forecast errors.** Knowledge about the future is seldom exact – as the most prominent example for forecasting uncertainty, weather conditions are well-known to have foiled plenty a schedule. They influence flight routes, driving speed, harvest quality, and many more aspects of everyday life. Other examples are demographic developments, or share prices.

4. **Changing environments due to long-term solutions.** When a problem solution is put into practice in a long-term setting, the environment naturally changes over the course of time. Timetables in public transport are one example,

in which the same schedule might need to serve for rain, snow, wind, low and high passenger demand, etc. Even though there might not be an error involved when these conditions are identified beforehand, still one solution must be able to suit very different scenarios.

In robust optimization – contrary to the setting of stochastic programming –, it is typically not assumed that any probability distribution for the uncertainty is known.

In many cases however, there exists a so-called *nominal scenario*. Depending on the uncertainty type, this may either be the coefficient of the given precision for numerical errors (1), the measured value for measurement errors (2), the most likely forecast for forecast errors (3), or an average environment for long-terms solutions (4).

Uncertain optimization problems. We consider optimization problems that can be written in the form

$$(P) \qquad \begin{aligned} \min\ & f(x) \\ \text{s.t. } & F(x) \le 0 \\ & x \in \mathcal{X}, \end{aligned}$$

where $F : \mathbb{R}^n \to \mathbb{R}^m$ describes the m problem constraints, $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, and $\mathcal{X} \subseteq \mathbb{R}^n$ is the variable space. In real-world applications, both the constraints and the objective may depend on parameters which are uncertain. In order to accommodate such uncertainties, instead of (P), the following parameterized *family* of problems is considered:

$$(P(\xi)) \qquad \begin{aligned} \min\ & f(x,\xi) \\ \text{s.t. } & F(x,\xi) \le 0 \\ & x \in \mathcal{X}, \end{aligned}$$

where $F(\cdot,\xi) : \mathbb{R}^n \to \mathbb{R}^m$ and $f(\cdot,\xi) : \mathbb{R}^n \to \mathbb{R}$ for any fixed $\xi \in \mathbb{R}^M$, which describes a *scenario* that may occur.

Although it is in practice often not known exactly which values such a scenario ξ may take for an optimization problem $P(\xi)$, we assume that it is known that ξ lies within a given *uncertainty set* $\mathcal{U} \subseteq \mathbb{R}^M$ that represents the scenarios we assume to be likely enough to be considered in our analysis.

The *uncertain optimization problem* corresponding to $P(\xi)$ is then denoted as

$$(P(\xi), \xi \in \mathcal{U})\,. \tag{1.1}$$

Note that the uncertain optimization problem in fact consists of a whole set of parameterized problems, that is often even infinitely large. The purpose of robust optimization models is to transform this family of problems into a single problem again. The resulting robust problem is called the *robust counterpart.*

The choice of the uncertainty set is of major impact not only on the type of robustness that we consider, but also on the computational complexity of the robustness models, and should be made carefully by the modeler. Also, the way the functions f and F depend on ξ leaves some freedom to the modeler's decision - in the simplest case, ξ coincides with the uncertain parameters of the given optimization problem.

Example 1.1. *As an example, consider a linear program* $\min\{c^t x : Ax \le b, x \in \mathbb{R}^n\}$ *with a coefficient matrix $A \in \mathbb{R}^{m\times n}$, a right-hand side vector $b \in \mathbb{R}^m$ and a cost vector $c \in \mathbb{R}^n$. If $A, b,$ and c are treated as uncertain parameters, we write*

$$\begin{aligned} P(A,b,c) \qquad & \min\ f(x,(A,b,c)) = c^t x \\ & s.t.\ F(x,(A,b,c)) = Ax - b \le 0 \\ & \quad x \in \mathbb{R}^n, \end{aligned}$$

i.e., $\xi = (A,b,c) \in \mathbb{R}^M$ with $M = n\cdot m + n + m$

However, it is also possible that the unknown parameters A, b, c may depend on (other) uncertain parameters $\xi \in \mathbb{R}^M$ where M need not be the number of uncertain parameters of the given problem. For example, there might be $M = 1$ parameter $\xi \in \mathbb{R}$ which determines all values of A, b, c. As an example imagine that the temperature determines the properties of different materials. In such a case we would have

$$\begin{aligned} f(x,\xi) &: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}, \text{ and} \\ F(x,\xi) &: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}, \end{aligned}$$

where $f(x,\xi) = c(\xi)^t x$ and $F(x,\xi) = A(\xi)x - b(\xi)$. It is also allowed that M is larger than the number of parameters used for modeling the optimization problem.

For a given uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$, we denote by

$$\mathcal{F}(\xi) = \{x \in \mathcal{X} : F(x,\xi) \le 0\}$$

the feasible set of scenario $\xi \in \mathcal{U}$. Furthermore, if there exists a nominal scenario, it is typically denoted by $\hat{\xi} \in \mathcal{U}$. The optimal objective value for a single scenario $\xi \in \mathcal{U}$ is denoted by $f^*(\xi)$.

We say that an uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ has convex (quasiconvex, affine, linear) uncertainty, when the function $F(x,\cdot)$ is convex (quasiconvex, affine, linear) for all $x \in \mathcal{X}$.

Common uncertainty sets. There are some types of uncertainty sets that are frequently used in current literature. These include:

1. Finite uncertainty $\mathcal{U} = \left\{\xi^1, \dots, \xi^N\right\}$

2. Interval-based uncertainty $\mathcal{U} = [\underline{\xi}_1, \overline{\xi}_1] \times \dots \times [\underline{\xi}_M, \overline{\xi}_M]$

3. Polytopic uncertainty $\mathcal{U} = \operatorname{conv}\left\{\xi^1, \dots, \xi^N\right\}$

4. Norm-based uncertainty $\mathcal{U} = \left\{\xi \in \mathbb{R}^M : \|\xi - \hat{\xi}\| \leq \alpha\right\}$
5. Ellipsoidal uncertainty $\mathcal{U} = \left\{\xi \in \mathbb{R}^M : \sqrt{\sum_{i=1}^M \xi_i^2/\sigma_i^2} \leq \Omega\right\}$
6. Constraint-wise uncertainty $\mathcal{U} = \mathcal{U}_1 \times \ldots \times \mathcal{U}_m$, where $\mathcal{U}_i$ only affects constraint i

where $\operatorname{conv}\left\{\xi^1, \ldots, \xi^N\right\} = \left\{\sum_{i=1}^N \lambda_i \xi^i : \sum_{i=1}^N \lambda_i = 1, \lambda \in \mathbb{R}_+^N\right\}$ denotes the convex hull of a set of points. Note that this classification is not exclusive, i.e., a given uncertainy set can belong to multiple types at the same time.

Example 1.2. *We consider the uncertain optimization problem*

$$\begin{aligned} P(\xi_1, \xi_2) \quad & \max\ x_1 + x_2 \\ & x_1 \leq \xi_1 \\ & x_2 \leq \xi_2 \\ & x_1, x_2 \in \mathbb{R}_+ \end{aligned}$$

with $(\xi_1, \xi_2) \in \mathcal{U} = [0,1] \times [0,1]$. *We can classify that the problem has an interval-based uncertainty, a polytopic uncertainty with respect to* $\{(0,0),(1,0),(0,1),(1,1)\}$, *a norm-based uncertainty with respect to* $\|\cdot\|_\infty$, *and a constraint-wise uncertainty.*

1.3 Methodology and Outline

Methodology. In this work we follow an application– and experiment–driven approach to robust optimization, motivated by the paradigm of *algorithm engineering*, as described in [San09] and [MHS10]. The outline is presented in Figure 1.1.

This algorithm engineering *cycle* can be interpreted in the following way: Concrete problem applications motivate the design of appropriate robustness models, which in turn spark a mathematical analysis that increases the problem knowledge. On the other hand, both model and analysis lead to algorithms, which are implemented in reusable program libraries, and used for experiments. The experimental results in turn show weak points of the model and the algorithm, and lead to a reconsideration of both. This mutual influence therefore creates *circles* of problem investigation, whose linearization form the basis for the structure of this work.

Generally speaking, this work lies in the fracture between *modeling* and *solving*. While modeling a problem brings applications into the domain of mathematics, an overemphasis of the theoretical analysis of models moves too far away from practice. Solving a model on the other hand pushes mathematics back into practice, but an overemphasis of this aspect cannot be effective (or science after all) without theoretical analysis. We focus on both theoretical and practical modeling and solving aspects trying to find a balance that combines the best of both worlds for robust optimization.

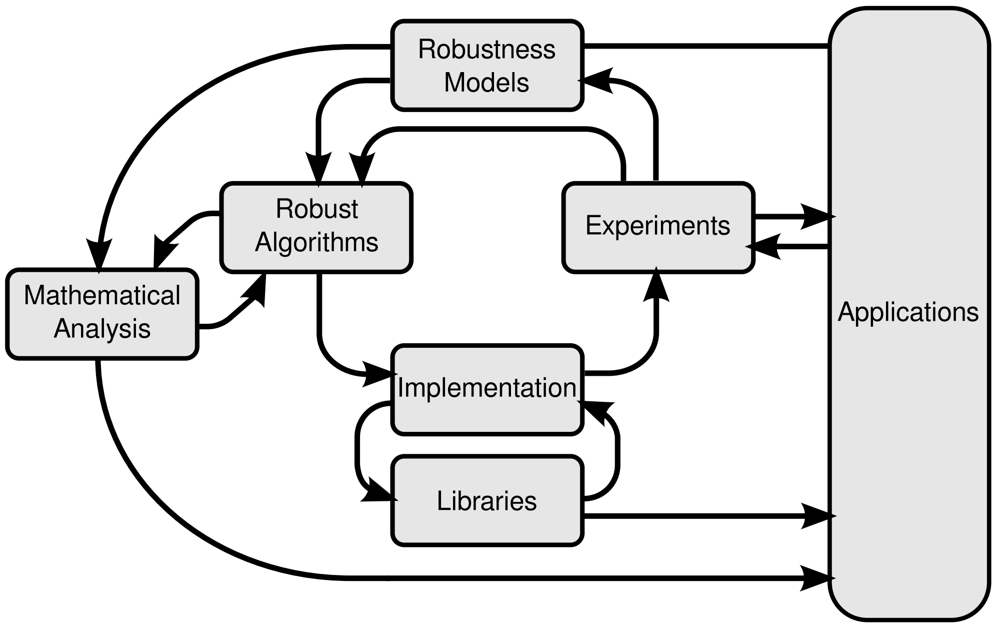


Figure 1.1 Methodological structure.

Outline. The remainder of this work is structured as follows:

- In Chapter 2, current approaches to robust optimization are discussed.
- We introduce the new robustness concepts of *RecFeas* and *RecOpt* in Chapter 3, and derive theoretical properties as well as relations to and between the current approaches to robustness as presented in Chapter 2.
- In the two following chapters, we turn our attention to applications in robust optimization. As solution approaches may highly differ, we separately consider continuous problem applications in Chapter 4, including linear programming and aperiodic timetabling, and discrete problem applications in Chapter 5, including loadplanning, Steiner trees, periodic timetabling and the timetable information problem.
- Algorithms need reusable software libraries to be brought into practice. Therefore, Chapter 6 describes the *Robust Optimization Programming Interface* (ROPI) that connects solver interfaces with robust optimization algorithms.
- Chapter 7 concludes this work with a discussion of the presented results and an outlook.
- In Chapter 8, the author's contribution to each part of this work is presented.

This implies the following relations between the chapters:

$$\begin{tikzcd}
 & & & \text{Ch.4} \arrow[dr] & & \\
\text{Ch.1} \arrow[r] & \text{Ch.2} \arrow[r] & \text{Ch.3} \arrow[ur] \arrow[dr] & & \text{Ch.6} \arrow[r] & \text{Ch.7} \\
 & & & \text{Ch.5} \arrow[ur] & &
\end{tikzcd}$$

Note that also the sections in Chapter 4 and Chapter 5 can be read in parallel by a multi-threaded reader – everybody else can at least skip sections here and still understand the rest of the work.

2 Literature Review on Robust Optimization Concepts

In this Chapter an overview to some of the current approaches to robust optimization is given. We begin with the basic concept of *strict robustness*, and proceed with the approach of *Bertsimas and Sim*, *adjustable*, *recovery* and *light robustness*, *min max regret* optimization, and shortly consider the concepts of *reliability*, *soft robustness*, *uncertainty feature optimization*, and *Mulvey et al.* We conclude this section with some words on the relation to stochastic optimization.

2.1 Strict Robustness

A solution $x \in \mathcal{X}$ to the uncertain problem $(P(\xi), \xi \in \mathcal{U})$ is called *strictly robust* if it is feasible for all scenarios in $\mathcal{U}$, i.e. if $F(x, \xi) \leq 0$ for all $\xi \in \mathcal{U}$. The objective usually follows the pessimistic view of minimizing the worst-case over all scenarios. Denoting the set of strictly robust solutions with respect to the uncertainty set $\mathcal{U}$ by

$$\mathrm{SR}(\mathcal{U}) = \bigcap_{\xi \in \mathcal{U}} \mathcal{F}(\xi),$$

the robust counterpart of the uncertain optimization problem is given as

$$\begin{array}{llrl} \text{(SR)} & \min & \sup\limits_{\xi \in \mathcal{U}} & f(x, \xi) \\ & & \text{s.t.} & x \in \mathrm{SR}(\mathcal{U}) \\ & & & x \in \mathcal{X}. \end{array}$$

The first to consider this type of problems from the perspective of *generalized linear programs* was Soyster [Soy73] for uncertainty sets $\mathcal{U}$ of type

$$\mathcal{U} = K_1 \times \ldots \times K_n,$$

where the set K_i contains possible column vectors A_i of the coefficient matrix A. In [Soy73] it is shown that (SR) can be formulated as a linear program if the sets K_i are compact and convex. Subsequent works on this topic include [Fal76] and [Thu80].

However, building this approach into a strong theoretic framework is due to a series of papers by Ben-Tal, Nemirovski and co-workers [BTN98, BTN99, BTN00]. A summary of their results can be found in the book [BTGN09]. They show that in case of Euclidean balls K_i, the resulting linear program can be explicitly stated, and further studied the theory of strict robustness for various other uncertainty sets. For polyhedral uncertainty sets $\mathcal{U} = \text{conv}\{\xi^1, \ldots, \xi^N\}$ with convex uncertainty in the objective and in the constraints, they show that the robust counterpart maintains many properties of the original program. For example, the strict robust counterpart (SR) of a linear program is again a linear program, and also differentiability and convexity are transferred from $P(\xi)$ to (SR). They furthermore investigate strict robustness for uncertainty sets $\mathcal{U}$ with constraint-wise uncertainty and with ellipsoidal uncertainty.

For infinite uncertainty sets, a scenario sampling approach has been analyzed in a series of papers [CC05, CC06, CG08, Cal10]. They show that for a single uncertain constraint

$$F(x, \xi) \leq 0$$

of a convex optimization problem, the probability of the violation event $V(x) = P\{\xi \in \mathcal{U} : F(x, \xi) > 0\}$ is bounded by

$$P(V(x^*) > \epsilon) \leq \sum_{i=0}^{n-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i},$$

where N is the sample size, and $x^* \in \mathbb{R}^n$ is the optimal solution with respect to the sampled scenarios. This result holds under the assumption that every subset of scenarios is feasible, and is independent of the probability distribution over $\mathcal{U}$. Also, generalizations of this result to multiple constraints are presented.

Due to the high conservatism of strict robustness, further research in robust optimization focused to a high degree on ways to relax this concept. We now describe some of these approaches.

2.2 Bertsimas and Sim

One possibility to overcome the conservatism of strict robustness is to shrink the uncertainty set $\mathcal{U}$. This has been conceptually introduced by Bertsimas and Sim in [BS04] for linear programming problems. Their main assumption is that it is unlikely that all coefficients of one constraint change simultaneously to their worst-case values, but only constantly many.

Considering a constraint of the form

$$a_1 x_1 + \ldots + a_n x_n \leq b$$

with an uncertainty $\mathcal{U} = \{a \in \mathbb{R}^n : a_i \in [\hat{a}_i - d_i, \hat{a}_i + d_i], i = 1, \ldots, n\}$, a solution x needs to fulfill

$$\sum_{i=1}^{n} \hat{a}_i x_i + \max_{\substack{S \subseteq \{1,\ldots,n\}, \\ |S| = \Gamma, t \in \{1,\ldots,n\} \setminus S}} \left\{ \sum_{i \in S} d_i y_i + (\Gamma - \lfloor \Gamma \rfloor) d_t y_t \right\} \leq b$$

$$\begin{aligned}&-y_i \le x_i \le y_i \qquad \forall i = 1, \dots, n\\ &y \ge 0\end{aligned}$$

for a given parameter $\Gamma \in [0, n]$ and some values y, which are introduced to model the absolute value of x. This means that only Γ many uncertain coefficients may deviate from their nominal value at the same time; but for all possible subsets a solution x needs to be feasible.

The authors show that this model can be linearized by using the dual of the inner maximization problem, which yields

$$\begin{aligned}&\sum_{i=1}^{n} \hat{a}_i x_i + z\Gamma + \sum_{i=1}^{n} p_i \le b\\ &z + p_i \ge d_i y_i \qquad \forall i = 1, \dots, n\\ &-y_i \le x_i \le y_i \qquad \forall i = 1, \dots, n\\ &p, y, z \ge 0.\end{aligned}$$

In [BS04], the authors further present analytical probability bounds of constraint violation and consider the *price of robustness*, i.e., the inferiority of the objective value when robustness is taken into account, compared to what can be achieved in the nominal model.

In [BS03], the authors apply this concept to discrete linear optimization problems, and show that the programs can be reformulated analogously. They furthermore consider combinatorial problems in which all decisions are binary and only the objective function is uncertain, and present a solution algorithm that solves the robust counterpart by solving a sequence of nominal problems. They apply this concept to the network flow problem. The approach to combinatorial optimization problems has been generalized in [Ata06] and [GST12]. Further applications include supply chain optimization [BT06].

2.3 Adjustable Robustness

Motivated by two-stage stochastic programming, *adjustable robustness* as introduced in [BTGGN03] decomposes the variables into two sets: The values for the *here-and-now variables* have to be found in the robust optimization algorithm while the decision about the *wait-and-see variables* can wait until the actual scenario $\xi \in \mathcal{U}$ becomes known.

We therefore assume that the variables $z = (x, y)$ are splitted into $x \in X \subseteq \mathbb{R}^{n_1}$ and $y \in Y \subseteq \mathbb{R}^{n_2}$ with $n_1 + n_2 = n$, where the variables x need to be determined before the scenarios $\xi \in \mathcal{U}$ becomes known, while the variables y may be determined after ξ has been realized. The uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ is rewritten as

$$\begin{aligned}P(\xi) \quad &\min f(x, y, \xi)\\ &F(x, y, \xi) \le 0\end{aligned}$$

$$(x, y) \in X \times Y.$$

When fixing the here-and-now variables, one has to make sure that for any possible scenario $\xi \in \mathcal{U}$ there exists $y \in Y$ such that (x, y, ξ) is feasible. The set of adjustable robust solutions is therefore given by

$$\begin{aligned}\mathrm{aSR} &= \{x \in X : \forall \xi \in \mathcal{U}\ \exists y \in Y \text{ s.t. } (x, y) \in \mathcal{F}(\xi)\}\\ &= \bigcap_{\xi \in \mathcal{U}} Pr_X(\mathcal{F}(\xi)),\end{aligned}$$

where for some set $A \subseteq X \times Y$, $Pr_X(A) = \{x \in X : \exists y \in Y \text{ s.t. } (x, y) \in A\}$ denotes the projection of A on X.

The worst case objective for some $x \in \mathrm{aSR}$ is given as

$$z^{\mathrm{aSR}}(x) = \sup_{\xi \in \mathcal{U}} \inf_{y:(x,y) \in \mathcal{F}(\xi)} f(x, y, \xi).$$

The adjustable robust counterpart is then given as

$$\min\{z^{\mathrm{aSR}}(x) : x \in \mathrm{aSR}\}.$$

Note that this setting is also useful if an uncertain problem with "helper variables" is considered. A constraint of the form

$$\sum_{i=1}^{n} a_i x_i \leq b$$

can be equivalently rewritten to the form

$$\begin{aligned}\sum_{i=1}^{n} a_i x_i + y = b\\ y \geq 0,\end{aligned}$$

but the strictly robust counterpart of both problem formulations has a different meaning. Considering y as an adjustable problem variable though, results in an equivalent problem as the strictly robust counterpart of the formulation without y.

The adjustable robust counterpart is in general computationally intractable. In the case of linear programs, it is shown in [BTGGN03] that for constraint-wise uncertainty, the adjustable robust counterpart is equivalent to the strictly robust counterpart. Furthermore, this is also the case when there is a compact set $\mathcal{Y}(x)$ for all feasible x such that $F(x, y, \xi) \leq 0$ implies $y \in \mathcal{Y}$. [TTT08] shows that the adjustable problem can be reduced in the case of polytopic uncertainty, and quasiconvexity in the objective function. In [Ter09], algorithms for the linear case are presented, including a cutting-plane approach. In [BTB08], the adjustable robust counterpart of conic quadratic optimization problems is considered. The uncertain network flow and design problem with uncertain demand is considered in [AZ07]. Further applications include circuit design [MSO06].

2.4 Light Robustness

The lightly robust counterpart of an uncertain optimization problem, as developed in [FM09] and [Sch10], requires a fixed nominal quality of the solution, and among all solutions satisfying this standard, the concept asks for the most "robust" one in the sense of constraint violation. Specifically, the lightly robust counterpart is of the following form:

$$\text{(LR)} \qquad \begin{aligned} \max\ & \sum_{i=1}^{m} w_i \gamma_i \\ \text{s.t.}\ & f(x,\hat{\xi}) \leq (1+\rho) f^*(\hat{\xi}) \\ & F(x,\xi) \leq \gamma \qquad \forall \xi \in \mathcal{U} \\ & x \in \mathcal{X}, \gamma \in \mathbb{R}^m, \end{aligned}$$

where w_i models a penalty weight for the violation of constraint i and ρ determines the available budget. This approach is also combinable with the constraint relaxation concept of Bertsimas and Sim.

Note that a constraint of the form $F(x,\xi) \leq 0$ is equivalent to a constraint $\lambda F(x,\xi) \leq 0$ for any $\lambda > 0$; therefore, the coefficients w_i play an important role in balancing the allowed violation of the given constraints.

The lightly robust approach has been applied to timetabling [FM09, FSZ09], and timetable information [GKMH⁺11].

2.5 Recovery Robustness

Similar to adjustable robustness, *recovery robustness*, which has been developed in [CDS⁺07, Sti08, LLMS09, DDN09] and has independently also been used in [EMS09], is a two-stage concept. Its basic idea is to allow a class of *recovery algorithms* $\mathcal{A}$ that can be used in case of a disturbance. A solution x is called *recovery robust* with respect to $\mathcal{A}$ if for any possible scenario $\xi \in \mathcal{U}$ there exists an algorithm $A \in \mathcal{A}$ such that A applied to the solution x and the scenario ξ constructs a solution $A(x,\xi) \in \mathcal{F}(\xi)$, i.e. a solution which is feasible for the current scenario.

Hence, the basic model is the following:

$$\text{(RR)} \qquad \begin{aligned} \min_{(x,A) \in \mathcal{F}(\hat{\xi}) \times \mathcal{A}}\ & f(x) \\ \text{s.t.}\ & A(x,\xi) \in \mathcal{F}(\xi)\ \forall \xi \in \mathcal{U}. \end{aligned}$$

It can be extended by including the recovery costs of a solution x: Let $d(A(x,\xi))$ be a possible vector-valued function that measures the costs of the recovery, and let $\lambda \in \Lambda$ be a limit on the recovery costs, i.e., $\lambda \geq d(A(x,\xi))$ for all $\xi \in \mathcal{U}$. Assume that there is some cost function $g : \Lambda \to \mathbb{R}$ associated with λ.

Setting

$$A(x,\xi,\lambda) \in \mathcal{F}'(\xi) \iff d(A(x,\xi)) \leq \lambda \ \wedge \ A(x,\xi) \in \mathcal{F}(\xi)$$

gives the recovery robust counterpart with limited recovery costs:

$$\text{(RR-LIM)} \qquad \min_{(x,A,\lambda)\in\mathcal{F}(\hat{\xi})\times\mathcal{A}\times\Lambda} \quad f(x)+g(\lambda) \qquad \text{s.t.} \quad A(x,\xi,\lambda)\in\mathcal{F}'(\xi)\ \forall\xi\in\mathcal{U}.$$

Due to the generality of this robustness model, the computational tractability heavily depends on the problem, the recovery algorithms and the uncertainty under consideration.

Applications include recovery robust models for linear programming [Sti08], shunting [CDS+07], timetabling [CDS+09b], platforming [CGST08], the empty repositioning problem [EMS09], railway rolling stock planning [CCG+12] and the knapsack problem [BKK11].

2.6 Regret Optimization

Regret optimization differs from the other presented robustness concepts insofar it only considers uncertainty in the objective function. Instead of minimizing the worst-case performance of a solution, it minimizes the difference to the objective function of the best solution that would have been possible in a scenario.

Let $f^*(\xi)$ denote the best objective value in scenario $\xi\in\mathcal{U}$. The min-max regret counterpart of an uncertain optimization problem with uncertainty in the objective is then given by

$$\begin{aligned} \text{(Regret)} \qquad & \min \sup_{\xi\in\mathcal{U}} \Big(f(x,\xi)-f^*(\xi)\Big) \\ & \text{s.t.} \quad F(x)\le 0 \\ & \qquad\ \ x\in\mathcal{X}. \end{aligned}$$

For a survey on this concept, see [ABV09] and [KY97].

If the original problem is polynomially solvable, there is an N-approximation algorithm for finite uncertainty sets [ABV09], where N is the number of scenarios, and a 2-approximation algorithm for interval-based uncertainty [KZ06b].

Due to its generality, applications of this concept to concrete problems are abundant. In [YKP01], the concept is applied to the spanning tree problem with edge weights from an interval-based uncertainty set, and further generalized to matroids in [YKP07]. Its approximation complexity for the shortest path, the spanning tree, and the knapsack problem is analyzed in [ABV05].

2.7 Further Robustness Concepts

Reliability. Another approach to robust optimization is to relax the constraints of strict robustness. This leads to the concept of *reliability* of Ben-Tal and Nemirovski

[BTN00], in which the constraints $F(x,\xi) \le 0$ are replaced by $F(x,\xi) \le \gamma$ for some $\gamma \in \mathbb{R}^m_{\ge 0}$. A solution x which satisfies

$$F(x,\xi) \le \gamma \text{ for all } \xi \in \mathcal{U}$$

is called *reliable with respect to* γ. The goal is to find a reliable solution which minimizes the original objective function in the worst case. Similar to light robustness, one has to be careful that the representation of the constraints does not affect the reliability of the solution, otherwise one may obtain the counter-intuitive result that, although the constraints $F(x,\xi) \le 0$ can also be written as $\Psi(F(x,\xi)) \le 0$ for any increasing Ψ with $\Psi(0) = 0$, what is understood by a robust solution may be different if one models the constraints with F or with $\Psi(F)$.

Soft Robustness. The basic idea of *soft robustness* as introduced in [BTBB10] is to handle the conservatism of the strict robust approach by considering a nested family of uncertainty sets, and allowing more deviation in the constraints for larger uncertainties. Specifically, instead of an uncertainty set $\mathcal{U} \subseteq \mathbb{R}^M$, a family of uncertainties $\{\mathcal{U}(\varepsilon) \subseteq \mathcal{U}\}_{\varepsilon>0}$ with $\mathcal{U}(\varepsilon_1) \subseteq \mathcal{U}(\varepsilon_2)$ for all $\varepsilon_2 \ge \varepsilon_1$ is used. The set of soft robust solutions is then given as

$$\mathrm{sR} = \{x \in \mathcal{X} : F(x,\xi) \le \varepsilon \ \forall \xi \in \mathcal{U}(\varepsilon),\ \varepsilon > 0\}\,.$$

Note that strict robustness is a special case with $\mathcal{U}(\varepsilon) = \mathcal{U}$ for all $\varepsilon > 0$.

In [BTBB10], the authors show that a solution to the soft robust problem counterpart can be found by solving a sequence of strict robust problems using a bisection approach over ε, and analyze the numerical performance on a bond portfolio and an asset allocation problem.

Uncertainty Feature Optimization. Instead of assuming that an explicit uncertainty set is given, which may be hard to model for real-world problems, the uncertainty feature optimization (UFO) approach [ESB11] rather assumes that the robustness of a solution is given by an explicit function. For an uncertain optimization problem $(P(\xi))$, let $\mu : \mathbb{R}^n \to \mathbb{R}^p$ be a measure for p robustness features. The UFO-counterpart of the uncertain problem is then given by

$$\text{(UFO)} \qquad \begin{aligned} \operatorname{vecmax}\ & \mu(x) \\ \text{s.t. }\ & F(x) \le 0 \\ & f(x) \le (1+\rho) f^*(\hat{\xi}) \\ & x \in \mathcal{X}, \end{aligned}$$

where $f^*(\hat{\xi})$ denotes the best objective value to the nominal problem. The authors can show that this approach generalizes both stochastic optimization and the approach of Bertsimas and Sim. In [Egg09], the UFO approach is applied to real-world airline scheduling problems.

Mulvey et al. Being actually a predecessor of the work of Ben-Tal et al, Mulvey, Vanderbei and Zenios introduced in [MVZ95] a framework for robust optimization of uncertain linear programs. We consider an uncertain optimization problem of the form

$$\begin{aligned} (\mathrm{P}(B,C,e)) \qquad & \min\ c^t x + d^t y \\ & \text{s.t. } Ax = b \\ & \quad Bx + Cy = e \\ & \quad x \in \mathbb{R}^{n_1}_+, y \in \mathbb{R}^{n_2}_+, \end{aligned}$$

where x represents a vector of *design* variables that cannot be adjusted, and y a vector of *control* variables that can be adjusted when the realized scenario becomes known. For a finite uncertainty set $\mathcal{U} = \{(B^1, C^1, e^1), \ldots, (B^N, C^N, e^N)\}$, the robust counterpart is given as

$$\begin{aligned} (\mathrm{Mul}) \qquad & \min\ \sigma(x, y^1, \ldots, y^N) + \omega\rho(z^1, \ldots, z^N) \\ & \text{s.t. } Ax = b \\ & \quad B^i x + C^i y^i + z^i = e^i \ \forall i = 1, \ldots, N \\ & \quad x \in \mathbb{R}^{n_1}_+, y^i \in \mathbb{R}^{n_2}_+, z^i \in \mathbb{R}^m. \end{aligned}$$

The variables z^i are introduced to measure the infeasibility in every scenario, i.e., the deviation from the right-hand side. The function σ represents the *solution robustness.* It can be modeled as a worst-case function of the nominal objective

$$\sigma(x, y^1, \ldots, y^N) = c^t x + \max_{i=1,\ldots,N} d^t y^i$$

or, when probabilities p^i are known, as an expected nominal objective. The function ρ on the other hand represents the *model robustness* and depends on the infeasibility of the uncertain constraints. Possible penalty functions are

$$\begin{aligned} \rho(z^1, \ldots, z^N) &= \sum_{i=1}^{N} p_i \sum_{j=1}^{m} \max\{0, z^i_j\} \\ \text{or } &= \sum_{i=1}^{N} p_i (z^i)^t z^i. \end{aligned}$$

As (Mul) is actually a bicriteria model, ω is used as a scalarization factor to combine both objectives.

2.8 Relation to Stochastic Optimization

Even though *stochastic optimization* considers a fundamentally different setting to robust optimization, some approaches can be easily transferred. For a discussion on the differences between both concepts, see e.g. [BTGN09].

Stochastic optimization is a well-established and thoroughly researched optimization method, and this paragraph can only highlight some differences and similarities. Introductions can be found in [BL97] or [KW94], amongst others.

As the main difference to robust optimization, the existence of a probability distribution P for the uncertainty set is assumed. Possible objective functions of stochastic programming include

- the expected value $E_P[f(x,\xi)]$,
- the expected utility $-E_P[u(f(x,\xi))]$,
- a risk measure objective $E_P[f(x,\xi)] + \lambda R_P[f(x,\xi)]$,
- a Markowitz model $E_P[f(x,\xi)] + \lambda \text{var}_P[f(x,\xi)]$,
- or a point estimate $\hat{\xi}_P$ of $\mathcal{U}$: $f(x,\hat{\xi}_P)$.

For uncertain constraints, possible stochastic reformulations include chance constraints

$$P(\xi : x \in \mathcal{F}(\xi)) \geq 1 - \epsilon,$$

which are in general computationally difficult, or individual probabilistic constraints

$$P(\xi : F_k(x,\xi) \leq 0) \geq 1 - \epsilon_k,$$

which are easier to handle. So-called *fat constraints* demand feasibility for all scenarios, and are therefore a direct connection to robust optimization.

As an example, the stochastic counterpart of an uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ for a finite uncertainty set $\mathcal{U} = \{\xi^1, \ldots, \xi^N\}$ with probability distribution $\{p_1, \ldots, p_N\}$ and a single uncertain constraint F can be given as

$$\begin{aligned} \min \ & \sum_{i=1}^{N} p_i f(x,\xi^i) \\ \text{s.t.} \ & F(x,\xi^i) \leq M_i z_i \\ & \sum_{i=1}^{N} p_i z_i \leq \epsilon, \end{aligned}$$

where the variables z are introduced to model chance constraints.

Note that a probability distribution might be difficult to obtain for real-world problems. Furthermore, minimizing the expected objective value makes sense for problems in which the solution is often evaluated, but might be unwanted for once-in-a-lifetime decisions like buying a life insurance, for which the robust optimization paradigm is a considerable alternative.

In [Dem91], the author considers solutions to the following problem:

$$\min \sum_{s=1}^{N} p_s \|f(x,\xi^s) - f^*(\xi^s)\|^2 + \sum_{s=1}^{N} p_s \| \max\{F(\xi^s), 0\}\|^2$$

$$x \in \mathcal{X},$$

i.e., he tries to find solutions that are "close" to feasibility and optimality in every scenario. To do so, every single scenario is solved to optimality first. In [RW91], a similar model for multi-stage processes is considered.

For uncertain optimization problems with infinite uncertainty sets, a solution approach is *Sample Average Approximation* (SAA), which uses a finite subset of scenarios instead. For all measurable functions f, convergence can be shown from the law of large numbers, i.e., the sampled objective value converges to the original objective value.

As a final remark, we consider the relationship between stochastic and robust optimization from the point of view of the uncertainty set. When determining the set of possibly occurring scenarios, the problem modeler always needs to draw a line somewhere - although a meteor impact devastating a track is possible, it is so unlikely that we may neglect its occurrence. In this sense, a robust solution that is feasible for *all* scenarios is only feasible for *all considered* scenarios. Therefore we may argue that a robust optimization problem has some kind of probability estimate included in its uncertainty set as well.

3 New Concepts and Relations

The following part considers two new approaches to robust optimization that draw from a location-theoretic point of view on recovery in the solution space: Recovery-to-Feasibility (RecFeas) in Section 3.1, and Recovery-to-Optimality (RecOpt) in Section 3.2. After a detailed discussion of these approaches we conclude this Chapter with some aspects of relations between robustness concepts in Section 3.3.

3.1 RecFeas

3.1.1 Introduction

In this section we propose a variation of recovery robustness based on geometric ideas, that is applicable for a wide range of problems. In particular, an optimal solution can be determined efficiently for linear programming problems for different types of uncertainties. For more complex settings reduction approaches are proposed.

This section is structured as follows: We introduce our model *recovery-to-feasibility* in Subsection 3.1.2, and analyze the recovery-robust counterpart for finite scenario sets in Subsection 3.1.3, and for infinite scenario sets in Subsection 3.1.4. In both subsections we derive exact solution approaches finding the best robust solution with respect to the recovery costs. We then turn our attention to algorithms for recovery-to-feasibility in Subsection 3.1.5: We consider an iterative approach for the case of a finite uncertainty set, and a sampling approach for the case of an infinite uncertainty set.

The section is concluded by a summary of our results.

3.1.2 A New Model: Recovery-to-Feasibility

The idea of our model is based on the concepts of recovery robustness, in particular on [LLMS09]. The main difference is that we replace the recovery algorithm by the properties of some norm (similar to the approach for shunting in [CCG$^+$12]) in order to obtain an approach that is easier to apply, as well as geometrically intuitive. We show that the complexity of the robust counterpart does not need to increase in our approach; in fact, the recovery-robust counterpart of a linear programming problem with polyhedral recovery costs stays a linear program. This even applies to infinite scenarios sets in some relevant cases. We start by introducing some notation.

Let a norm $\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$ be given. $\| \cdot \|$ should be chosen in such a way that the induced metric represents the time or the costs needed to change the solution $x \in \mathbb{R}^n$ into another solution $y \in \mathbb{R}^n$ (e.g. by a given recovery algorithm). Norms are frequently used in applications to model the recovery costs, e.g., in timetabling [LLMS09] or in recovery robust linear programming [Sti08]. Let $d(x, y) = \|y - x\|$ denote the distance induced by the norm $\| \cdot \|$. For a closed non-empty set $\mathcal{F} \subseteq \mathbb{R}^n$ we define the distance $d(x, \mathcal{F})$ from a point x to the set $\mathcal{F}$ as

$$d(x, \mathcal{F}) = \min_{y \in \mathcal{F}} d(x, y),$$

where the minimum exists.

Now let an uncertain optimization problem $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ including the set $\mathcal{X} \subseteq \mathbb{R}^n$ of implementable decisions be given. In the recovery-robust counterpart RecFeas($\mathcal{U}$), we ask for a solution $x \in \mathcal{X}$ which can be recovered with lowest possible costs in the worst case. For a recovery robust solution $x \in \mathcal{X}$ we hence require that for any scenario $\xi \in \mathcal{U}$, it can be transformed into another solution $y = y(x, \xi)$ which is feasible for ξ (i.e. $y \in \mathcal{F}(\xi)$), and we minimize the costs $d(x, y)$ of this recovery in the worst case. Formally, given an uncertain optimization problem $(\mathrm{P}(\xi), \xi \in \mathcal{U})$, a distance function d and a set $\mathcal{X}$, we define the optimization problem

$$\text{RecFeas}(\mathcal{U}) \qquad \min_{x \in \mathcal{X}} \sup_{\xi \in \mathcal{U}} d(x, \mathcal{F}(\xi)).$$

If not specified otherwise, d is induced by an arbitrary norm and $\mathcal{X} = \mathbb{R}^n$. The set $\mathcal{X}$ of implementable decisions can be used to model different important issues. Indeed, $\mathcal{X}$ may refer to technological or physical constraints on the variables (e.g. some variables represent non-negative magnitudes, or there are precedence constraints between two events), or may refer to modeling constraints (e.g. some variables are Boolean, and thus they can only take the values 0 and 1). The inclusion of constraints makes also sense in applications where a nominal scenario $\hat{\xi}$ is given and the solution x is required to be feasible for this scenario, or $\mathcal{X}$ can (similar to light robustness) define a minimal nominal standard of the robust solution x. In order to avoid non-existence of an optimal solution we always assume that $\mathcal{X}$ is closed.

Let us denote the objective function of RecFeas($\mathcal{U}$) by

$$r(x,\mathcal{U}) = \sup_{\xi \in \mathcal{U}} d(x, \mathcal{F}(\xi))$$

and let us call $r(x,\mathcal{U})$ the (recovery) radius of x with respect to $\mathcal{U}$. Let $r^*(\mathcal{U})$ denote the best possible recovery radius (if it exists). Any optimal solution x^* with

$$r(x^*,\mathcal{U}) = r^*(\mathcal{U})$$

is called a *center* with respect to $\{\mathcal{F}(\xi) : \xi \in \mathcal{U}\}$. We may omit $\mathcal{U}$ in both $r(x.\mathcal{U})$ and RecFeas($\mathcal{U}$) if it is clear from the context which uncertainty set is meant.

The terminology used here is inspired by the connection of our recovery-to-feasibility model to some location problems. In a classical location problem (known as the *Weber problem* or as the *Fermat-Torricelli problem*, see e.g. [DKSW01]) we have given a set of points, called *existing facilities*, and we look for a new point minimizing the distances to these given points. If the distance to the farthest point is considered as the objective function, the problem is called the *center location problem* [PC12]. The recovery-robust counterpart RecFeas($\mathcal{U}$)

$$\min_{x\in\mathcal{X}} \max_{k=1,\dots,N} d(x,\mathcal{F}(\xi^k))$$

of an uncertain optimization problem with a finite uncertainty set $\mathcal{U}$ can hence be interpreted as a location problem in which the existing facilities are not points but the sets $\mathcal{F}(\xi^1),\dots,\mathcal{F}(\xi^N)$. RecFeas($\mathcal{U}$) asks for a point $x \in \mathcal{X}$ which minimizes the maximum distance to the given sets. The notation of location theory is adapted by calling such a point – which then is an optimal solution to RecFeas($\mathcal{U}$) – a *center* of $\mathcal{F}(\xi^1),\dots,\mathcal{F}(\xi^N)$.

The connection between RecFeas($\mathcal{U}$) and (point) location problems becomes clearer for specific shapes of the sets $\mathcal{F}(\xi)$. For instance, let the sets $\mathcal{F}(\xi)$ be scaled and translated unit balls of the norm $\|\cdot\|$, i.e.

$$\mathcal{F}(\xi) = \{y \in \mathbb{R}^n : d(y, c(\xi)) \le r\}$$

for some $c(\xi) \in \mathbb{R}^n$ for all $\xi \in \mathcal{U}$ and $r \in \mathbb{R}_{\ge 0}$. In this case we obtain that

$$d(x,\mathcal{F}(\xi)) = \begin{cases} d(x,c(\xi)) - r & \text{if } d(x,c(\xi)) > r \\ 0 & \text{if } d(x,c(\xi)) \le r \end{cases}$$

and it turns out that the center of the location problem with existing (point) facilities $\{c(\xi) : \xi \in \mathcal{U}\}$ is just an optimal solution to RecFeas($\mathcal{U}$).

Note that, instead of minimizing the maximum distance to all feasible sets, we may also consider the problem of minimizing the sum of distances to all sets, which is known as the *median problem*. However, in this section we focus on the center problem.

Let us now describe some general properties of problem RecFeas($\mathcal{U}$). First observe that, even when the function F has a very simple form, the set $\mathcal{F}(\xi)$ may collapse from being a very large set to become empty. For instance, for $F(x,\xi) = 1 - x\xi$, we have that $\mathcal{F}(\xi) = [\frac{1}{\xi}, +\infty)$ for $\xi > 0$, $\mathcal{F}(0) = \emptyset$, and $\mathcal{F}(\xi) = (-\infty, \frac{1}{\xi}]$ for $\xi < 0$.

Since d is induced by a norm we know that

$$0 \le r(x,\mathcal{U}) \le +\infty \quad \text{for all } x \in \mathbb{R}^n, \tag{3.1}$$

thus the optimal value of RecFeas($\mathcal{U}$) is bounded by zero from below, but is $+\infty$ if all points x have infinite radius $r(x,\mathcal{U})$. This event may happen even when the sets $\mathcal{F}(\xi)$ are always non-empty. Indeed, consider, for instance, $\mathcal{X} = \mathbb{R}$, $\mathcal{F}(\xi) = \{\xi\}$ for all $\xi \in \mathcal{U} = \mathbb{R}$.

One has, however, that finiteness of $r(x,\mathcal{U})$ at one point implies finiteness everywhere, and, in that case, the radius is Lipschitz-continuous, as shown in the following result.

Lemma 3.1. *Let an uncertain optimization problem* $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ *be given and let* d *be induced by a norm. Suppose there exists* $x_0 \in \mathbb{R}^n$ *such that* $r(x_0, \mathcal{U}) < +\infty$*. Then,* $r(x, \mathcal{U}) < +\infty$ *for all* $x \in \mathbb{R}^n$*. In such a case, the function* $x \in \mathbb{R}^n \longmapsto r(x, \mathcal{U})$ *is Lipschitz-continuous with Lipschitz constant* $L = 1$*.*

Proof. Take $x \in \mathbb{R}^n$ and $\xi \in \mathcal{U}$. Let $y \in \mathcal{F}(\xi)$ such that $d(x_0, \mathcal{F}(\xi)) = d(x_0, y)$. We have that

$$\begin{aligned} d(x, \mathcal{F}(\xi)) &\leq d(x, y) \\ &\leq d(x, x_0) + d(x_0, y) = d(x, x_0) + d(x_0, \mathcal{F}(\xi)) \end{aligned}$$

Hence,

$$\max_{\xi \in \mathcal{U}} d(x, \mathcal{F}(\xi)) \leq d(x, x_0) + \max_{\xi \in \mathcal{U}} d(x_0, \mathcal{F}(\xi)) < +\infty.$$

Consequently, $r(x, \mathcal{U})$ is finite everywhere.

We now show that the function is also Lipschitz-continuous. Let $\varepsilon > 0$, and let $x, x' \in \mathbb{R}^n$. Take ξ^* such that

$$\varepsilon + d(x, \mathcal{F}(\xi^*)) \geq r(x, \mathcal{U}).$$

Since $\mathcal{F}(\xi^*)$ is closed, take also $y' \in \mathcal{F}(\xi^*)$ such that $d(x', \mathcal{F}(\xi^*)) = d(x', y')$.

Then,

$$\begin{aligned} r(x, \mathcal{U}) - r(x', \mathcal{U}) &\leq \varepsilon + d(x, \mathcal{F}(\xi^*)) - d(x', \mathcal{F}(\xi^*)) \\ &\leq \varepsilon + d(x, y') - d(x', y') \\ &\leq \varepsilon + d(x, x'). \end{aligned}$$

Since this inequality holds for any $\varepsilon > 0$, we obtain

$$r(x, \mathcal{U}) - r(x', \mathcal{U}) \leq d(x, x'),$$

hence the function $r(\cdot, \mathcal{U})$ is Lipschitz-continuous with Lipschitz constant 1. □

In what follows we assume finiteness of the optimal value of RecFeas($\mathcal{U}$), and thus Lipschitz-continuity of $r(\cdot, \mathcal{U})$. Hence, RecFeas($\mathcal{U}$) may be solved by using standard Lipschitz optimization methods [SK10].

A finite optimal value does not guarantee the existence of optimal solutions, even if $\mathcal{U}$ is finite. This is shown in the following example:

Example 3.2. *Consider the uncertain program*

$$\begin{array}{lrl} P(\xi) & \min & f(x_1, x_2) \\ & s.t. & 1 \leq \xi x_1 x_2 \\ & & \xi x_1 \geq 0 \\ & & x_2 \geq 0, \end{array}$$

where $\mathcal{U} = \{-1, 1\}$ is the uncertainty set, $f(x_1, x_2)$ is any objective function and $\mathcal{X} = \mathbb{R}$. The feasible sets of scenario $\xi_1 = -1$ and scenario $\xi_2 = 1$ are given by:

$$\begin{aligned}\mathcal{F}(-1) &= \{(x_1, x_2) \in \mathbb{R}^2 : x_1x_2 \le -1,\ x_1 \le 0,\ x_2 \ge 0\},\\ \mathcal{F}(1) &= \{(x_1, x_2) \in \mathbb{R}^2 : x_1x_2 \ge 1,\ x_1, x_2 \ge 0\}.\end{aligned}$$

Both feasible sets are depicted in Figure 3.1. Since the sets have a common point at infinity, it follows that

$$\inf_{x\in\mathbb{R}^2} r(x,\mathcal{U}) = 0,$$

but $r(x,\mathcal{U}) > 0$ for all $x \in \mathbb{R}^2$. Hence, an optimal solution to RecFeas($\mathcal{U}$) *does not exist.*

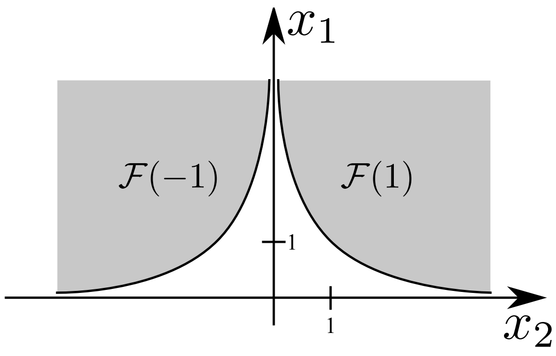


Figure 3.1 An example where an optimal solution to RecFeas($\mathcal{U}$) does not exist, even for an uncertainty set $\mathcal{U}$ with only two scenarios.

For a given $x \in \mathbb{R}^n$ let us call $\xi \in \mathcal{U}$ a *worst-case scenario* with respect to x (and $\mathcal{U}$) if

$$d(x, \mathcal{F}(\xi)) = r(x,\mathcal{U})$$

and let $WC(x,\mathcal{U})$ be the set of all worst-case scenarios, i.e. scenarios $\xi \in \mathcal{U}$ yielding the maximal recovery distance for the solution x. Under certain assumptions, optimal solutions x^* have a set of worst-case scenarios $WC(x^*,\mathcal{U})$ with at least two elements, as shown in the following result.

Lemma 3.3. *Let an uncertain optimization problem* (P(ξ), $\xi \in \mathcal{U}$) *be given. Suppose that $\mathcal{U}$ is finite, $\mathcal{X} = \mathbb{R}^n$, and d is induced by a norm $\|\cdot\|$ and* RecFeas($\mathcal{U}$) *attains its optimum at some $x^* \in \mathbb{R}^n$. Then, $|WC(x^*,\mathcal{U})| \ge 2$.*

Proof. Finiteness of $\mathcal{U}$ implies that the maximum of $d(x^*, \mathcal{F}(\xi))$ must be attained at some ξ. Hence, $|WC(x^*,\mathcal{U})| \ge 1$. In the case that $WC(x^*,\mathcal{U}) = \{\xi^*\}$ for only one scenario $\xi^* \in \mathcal{U}$, we could move x towards $\mathcal{F}(\xi)$ and shrink r to obtain a better solution.

Indeed, take $y^* \in \mathcal{F}(\xi^*)$ such that $d(x^*, y^*) = d(x^*, \mathcal{F}(\xi^*))$, and, for $\lambda \in [0,1]$, define x_λ as

$$x_\lambda = (1-\lambda)x^* + \lambda y^*.$$

Since, by assumption, $WC(x^*,\mathcal{U}) = \{\xi^*\}$ and $\mathcal{U}$ is finite, there exists $\varepsilon > 0$ such that

$$d(x^*, \mathcal{F}(\xi)) < d(x^*, \mathcal{F}(\xi^*)) - \varepsilon \qquad \forall \xi \in \mathcal{U},\, \xi \neq \xi^*.$$

Let us show that, for λ close to zero, x_λ has a strictly better objective value than x^*, which would be a contradiction. First we have

$$\begin{array}{rcl} d(x_\lambda, \mathcal{F}(\xi^*)) & \leq & d(x_\lambda, y^*) \\ & = & (1-\lambda)\|x^* - y^*\| = (1-\lambda) d(x^*, \mathcal{F}(\xi^*)) \\ & < & d(x^*, \mathcal{F}(\xi^*)) \text{ for } \lambda > 0. \end{array}$$

For the remaining scenarios $\xi \neq \xi^*$,

$$\begin{array}{rcl} d(x_\lambda, \mathcal{F}(\xi)) & \leq & \inf\limits_{y \in \mathcal{F}(\xi)} (\|x_\lambda - x^*\| + \|x^* - y\|) \\ & = & \lambda \|x^* - y^*\| + d(x^*, \mathcal{F}(\xi)) \\ & < & \lambda \|x^* - y^*\| + d(x^*, \mathcal{F}(\xi^*)) - \varepsilon \\ & < & d(x^*, \mathcal{F}(\xi^*)) \text{ for } \lambda < \dfrac{\varepsilon}{\|x^* - y^*\|}. \end{array}$$

Hence, for $0 < \lambda < \frac{\varepsilon}{\|x^*-y^*\|}$, we would have that

$$\max_{\xi \in \mathcal{U}} d(x_\lambda, \mathcal{F}(\xi)) < d(x^*, \mathcal{F}(\xi^*)) = \max_{\xi \in \mathcal{U}} d(x^*, \mathcal{F}(\xi)),$$

contradicting the optimality of x^*. □

If the finiteness assumption of Lemma 3.3 is dropped, not much can be said about the cardinality of $WC(x,\mathcal{U})$, since this set can be empty or a singleton:

Example 3.4. *Let* $\mathcal{U} = \{-1, 1\} \times [1, \infty)$, *and let* $F(x, (\xi_1, \xi_2)) = (x - \xi_1)(\xi_2 x - \xi_1 \xi_2 + \xi_1)$. *It is easily seen that*

$$\begin{array}{rcl} \mathcal{F}(-1, \xi_2) & = & [-1, -1 + \frac{1}{\xi_2}] \\ \mathcal{F}(1, \xi_2) & = & [1 - \frac{1}{\xi_2}, 1] \end{array} \tag{3.2}$$

For $x = 0$, $r(x,\mathcal{U}) = 1$, *but there is no* $\xi \in \mathcal{U}$ *with* $d(x, \mathcal{F}(\xi)) = 1$. *In other words,* $WC(0,\mathcal{U}) = \emptyset$.

If we slightly modify the definition of $\mathcal{U}$ *above, by considering*

$$\mathcal{U} = (\{-1, 1\} \times [1, \infty)) \cup \{(-1, 0)\},$$

and define $F(x, (\xi_1, \xi_2))$ *as*

$$F(x, (\xi_1, \xi_2)) = \begin{cases} (x - \xi_1)(\xi_2 x - \xi_1 \xi_2 + \xi_1), & \text{if } \xi_2 \neq 0 \\ (x - \xi_1)^2, & \text{if } \xi_2 = 0 \end{cases}$$

we have that $\mathcal{F}(\xi_1, \xi_2)$ *is defined as in (3.2) for* $(\xi_1, \xi_2) \in \{-1, 1\} \times [1, \infty)$, *and* $\mathcal{F}(-1, 0) = \{-1\}$. *Hence,* $WC(0,\mathcal{U}) = \{(-1, 0)\}$.

We close this section by pointing out the relation between recovery-to-feasibility and the concept of strict robustness of [BTGN09]. To this end note that

$$\mathrm{SR}(\mathcal{U}) = \bigcap_{\xi \in \mathcal{U}} \mathcal{F}(\xi)$$

is the set of strictly robust solutions of $(P(\xi), \xi \in \mathcal{U})$.

Lemma 3.5. *Let an uncertain problem* $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ *be given, let* $\mathcal{X} = \mathbb{R}^n$ *and* d *be induced by a norm. Let* $\mathrm{SR}(\mathcal{U})$ *be the set of strictly robust solutions and* $\mathrm{RecFeas}(\mathcal{U})$ *be the recovery-robust counterpart. Then the following holds:*

$$\mathrm{SR}(\mathcal{U}) \neq \emptyset \text{ if and only if } r^*(\mathcal{U}) = 0.$$

In particular, if $\mathrm{SR}(\mathcal{U}) \neq \emptyset$*, the optimal recovery radius exists and any* $x \in \mathrm{SR}(\mathcal{U})$ *is an optimal solution to* $\mathrm{RecFeas}(\mathcal{U})$.

Proof. Let $x \in \mathrm{SR}(\mathcal{U})$. Then $x \in \mathcal{F}(\xi)$ for all $\xi \in \mathcal{U}$. This means that $d(x, \mathcal{F}(\xi)) = 0$ for all $\xi \in \mathcal{U}$. From (3.1) we conclude $r(x,\mathcal{U}) = 0$, hence x is an optimal solution to $\mathrm{RecFeas}(\mathcal{U})$.

On the other hand, $r(x,\mathcal{U}) = 0$ implies $d(x, \mathcal{F}(\xi)) \leq 0$ for all $\xi \in \mathcal{U}$. Since, by assumption, $\mathcal{F}(\xi)$ is closed, this means that $x \in \mathcal{F}(\xi)$ for all $\xi \in \mathcal{U}$, hence the set of strictly robust solutions is not empty. □

A relation between recovery-to-feasibility and reliability (as in [BTN00]) is mentioned in Theorem 3.14. For a more detailed discussion of relations to robustness concepts from the current literature, we refer to Section 3.3.1.

In the next two sections we investigate recovery-to-feasibility for two cases: In Section 3.1.3 we look at problems with a finite uncertainty set $\mathcal{U}$. Particular problems with convex and with linear constraints are considered. In Section 3.1.4 we analyze the more general case of uncertainty sets containing an infinite number of scenarios. Our goal in this section is to identify special cases in which the infinite set $\mathcal{U}$ may be reduced to a finite set such that the approaches of Section 3.1.3 can be applied.

3.1.3 Recovery-to-Feasibility for a Finite Uncertainty Set

In this section we assume that $\mathcal{U}$ is finite, $\mathcal{U} = \{\xi^1, \ldots, \xi^N\}$. This simplifies the analysis, since we can explicitly search for a solution y^k for every scenario $\xi^k \in \mathcal{U}$. Using the y^k as variables we may formulate $\mathrm{RecFeas}(\mathcal{U})$ as

$$\begin{array}{lll} \min & r & \\ \text{s.t.} & F(y^k, \xi^k) \leq 0 & \text{for all } k = 1, \ldots, N \\ & d(x, y^k) \leq r & \text{for all } k = 1, \ldots, N \\ & x \in \mathcal{X}, r \in \mathbb{R} & \\ & y^k \in \mathcal{X} & \text{for all } k = 1, \ldots, N. \end{array} \tag{3.3}$$

Still such an optimization problem may be rather hard to solve because of the constraints defined by F and the possible nonlinearity of the distance constraint. Assuming that the distance used is the Euclidean $d^2(\cdot,\cdot)$, we can write RecFeas($\mathcal{U}$) equivalently as

$$\min_{x\in\mathcal{X}} \max_{1\leq k\leq N} d^2(x,\mathcal{F}(\xi^k)). \tag{3.4}$$

The function $x \longmapsto \max_k d^2(x,\mathcal{F}(\xi^k))$ is known to be *d.c.*, i.e., it can be written as a difference of two convex functions, and then the powerful tools of d.c. programming may be used to find a globally optimal solution if RecFeas($\mathcal{U}$) is low-dimensional [BC09, BCH09], or to design heuristics for more general cases [AT05].

Convex programming problems. We start with optimization problems P(ξ) that have convex feasible sets $\mathcal{F}(\xi)$ for all $\xi \in \mathcal{U}$. This is the case if the function F in the constraint

$$F(x,\xi) \leq 0$$

of P(ξ) is such that $F(\cdot,\xi) : \mathbb{R}^n \to \mathbb{R}^m$ is quasiconvex for all scenarios ξ.

Let us fix ξ. Let the distance d be induced by a norm. Then $d(\cdot,\mathcal{F}(\xi)) : \mathbb{R}^n \to \mathbb{R}$ describes the distance between a point and a convex set and hence is a convex function. We conclude that $r(x,\mathcal{U})$ is convex as the maximum of a finite set of convex functions.

Lemma 3.6. *Let d be convex, let $\mathcal{U}$ be finite and let $\mathcal{F}(\xi)$ be convex and closed for all $\xi \in \mathcal{U}$. Then, $r(\cdot,\mathcal{U})$ is a convex function.*

Proof. By definition,

$$r(x,\mathcal{U}) = \max_{i=1,\dots,N} d(x,\mathcal{F}(\xi^i))$$

We consider the functions

$$f_i(x) := \min_{y\in\mathcal{F}(\xi^i)} d(x,y),\ i = 1,\dots,N.$$

For a fixed $i \in \{1,\dots,N\}$, let x_1, x_2 and $\lambda \in [0,1]$ be given. Let $y_1 \in \arg\min d(x_1,y)$, $y_2 \in \arg\min d(x_2,y)$. Then

$$\begin{aligned}
& f_i(\lambda x_1 + (1-\lambda)x_2) \\
= & \min_{y\in\mathcal{F}(\xi^i)} d(\lambda x_1 + (1-\lambda)x_2, \mathcal{F}(\xi^i)) \\
\leq & \ d(\lambda x_1 + (1-\lambda)x_2, \lambda y_1 + (1-\lambda)y_2), \text{ as } \mathcal{F}(\xi^i) \text{ is convex} \\
\leq & \ \lambda d(x_1,y_1) + (1-\lambda)d(x_2,y_2), \text{ as } d \text{ is convex}
\end{aligned}$$

Therefore, each of the functions f_i is convex, and as the maximum of convex functions is convex, so is $r(\cdot,\mathcal{U})$. □

Lemma 3.7. *Consider an uncertain optimization problem* $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ *with quasiconvex* $F(\cdot, \xi)$ *for any fixed* ξ. *Let* $\mathcal{U}$ *be a finite set and* $\mathcal{X} \subseteq \mathbb{R}^n$ *be closed and convex. Let* d *be induced by a norm. Then problem (3.3) is a convex optimization problem.*

In order to solve RecFeas($\mathcal{U}$) one can hence apply algorithms suitable for convex programming, e.g. subgradient or bundle methods [SY06, HUL93]. In particular, since RecFeas($\mathcal{U}$) is unconstrained, a necessary and sufficient condition for a point x^* to be an optimal solution is

$$0 \in \partial(r(x^*,\mathcal{U})),$$

i.e., if 0 is contained in the subdifferential of r at the point x^*. By construction of $r(\cdot,\mathcal{U})$, we obtain

$$0 \in \operatorname{conv}\{\partial d(x^*, \mathcal{F}(\xi)) : \xi \in WC(x^*,\mathcal{U})\}$$

where $WC(x^*,\mathcal{U})$ is the set of worst-case scenarios [HUL93].

Now, $\partial d(x^*, \mathcal{F}(\xi))$ can be written in terms of the subdifferential of the distance used, see [CF02], where also easy representations for polyhedral norms or the Euclidean norm are presented. Although we do not know much a priori about the number of worst-case scenarios, we do not need to investigate all possible subsets but may restrict our search to sets which do not have more than $n + 1$ elements. This may be helpful in problems with a large number of scenarios but low dimension n for the decisions.

Theorem 3.8. *Let* $\mathcal{U}$ *be finite with cardinality of at least* $n + 1$. *Let* $\mathcal{X} = \mathbb{R}^n$ *and let* d *be induced by a norm. Suppose* RecFeas($\mathcal{U}$) *attains its optimum at some* x^*, *and that for each* ξ, *the function* $F(\cdot, \xi)$ *is quasiconvex. Then there exists a subset* $\bar{\mathcal{U}} \subseteq \mathcal{U}$ *of scenarios with* $2 \le |\bar{\mathcal{U}}| \le n + 1$ *such that*

$$r(\mathcal{U}) = r(x^*,\mathcal{U}) = r(x^*,\bar{\mathcal{U}}) = r(\bar{\mathcal{U}}).$$

Proof. Let x^* be optimal for RecFeas($\mathcal{U}$). The result is trivial if $r(x^*,\mathcal{U}) = 0$: take any collection of $n + 1$ scenarios. Accordingly, we may assume $r(x^*,\mathcal{U}) > 0$, which implies that x^* does not belong to all sets $\mathcal{F}(\xi)$.

By Lemma 3.3, $|WC(x^*,\mathcal{U})| \ge 2$. If $|WC(x^*,\mathcal{U})| \le n + 1$, then we are done. Otherwise, $|WC(x^*,\mathcal{U})| > n + 1$, we have by the optimality of x^* and convexity of the functions $d(\cdot, \mathcal{F}(\xi))$, that

$$0 \in \operatorname{conv}\{\partial d(x^*, \mathcal{F}(\xi)) : \xi \in WC(x^*,\mathcal{U})\}$$

By Carathéodory's theorem, $WC(x^*,\mathcal{U})$ contains a subset $\bar{\mathcal{U}}$, $1 \le |\bar{\mathcal{U}}| \le n+1$ such that $0 \in \operatorname{conv}\left\{\partial d(x^*, \mathcal{F}(\xi)) : \xi \in \bar{\mathcal{U}}\right\}$. Such $\bar{\mathcal{U}}$ clearly satisfies the conditions stated. □

Problems with linear constraints and polyhedral norms as recovery costs. As in the section before, we assume to be given a finite uncertainty set $\mathcal{U} = \{\xi^1, \dots, \xi^N\}$. Let us now consider the case that all sets $\mathcal{F}(\xi^k)$, $k = 1, \dots, N$ are polyhedral sets. More precisely, we consider problems of the type

$$\mathrm{P}(\xi) \qquad \min f(x, \xi)$$

$$\begin{aligned} \text{s.t. } & F(x,\xi) := A(\xi)x - b(\xi) \le 0 \\ & x \in \mathcal{X} \end{aligned}$$

with a finite uncertainty set $\mathcal{U} = \{\xi^1, \ldots, \xi^N\}$, linear constraints for every $\xi \in \mathcal{U}$ and a polyhedron $\mathcal{X}$.

Furthermore, let us assume that the distance d is induced by a block norm $\|\cdot\|$, i.e. a norm for which the unit ball is a polytope, see [WWR85, Wit64]. The most prominent examples for block norms are the Manhattan (ℓ_1) and the maximum (ℓ_∞) norm, which both may be suitable to represent recovery costs. In the case that the recovery costs are obtained by adding single costs of each component, the Manhattan norm is the right choice. The maximum norm may represent the recovery time in the case that a facility has to be moved along each coordinate (or a schedule has to be updated by a separate worker in every component) and the longest time determines the time for the complete update.

We also remark that it is possible to approximate any given norm arbitrarily close by block norms, since the class of block norms is a dense subset of all norms, see [WWR85]. Thus, the restriction to the class of block norms may not be a real restriction in a practical setting.

The goal of this section is to show that under the assumptions above, the recovery-robust counterpart RecFeas($\mathcal{U}$) of the uncertain optimization problem (P(ξ), $\xi \in \mathcal{U}$) is a linear program and a robust solution can hence be efficiently computed.

We start with some notation. Given a norm $\|\cdot\|$, let

$$B = \{x \in \mathbb{R}^n : \|x\| \le 1\}$$

denote its unit ball. Recall that the unit ball of a block norm $\|\cdot\|$ is a full-dimensional convex polytope which is symmetric with respect to the origin. Since such a polytope has a finite number S of extreme points, we may denote in the following the extreme points of B as

$$\mathrm{Ext}(B) = \{e_i : 1 \le i \le S\}.$$

Since B is symmetric with respect to the origin, $S \in \mathbb{N}$ is always an even number and for any $e_i \in \mathrm{Ext}(B)$ there exists another $e_j \in \mathrm{Ext}(B)$ such that $e_i = -e_j$.

The following property is crucial for the linear programming formulation of RecFeas($\mathcal{U}$). It shows that it is sufficient to consider only the extreme points Ext(B) of a block norm with unit ball B in order to compute $\|x\|$ for any point $x \in \mathbb{R}^n$.

Lemma 3.9 ([WWR85])**.** *A block norm $\|\cdot\|$ has a characterization as*

$$\|x\| = \min\left\{\sum_{i=1}^{S} \beta_i : x = \sum_{i=1}^{S} \beta_i e_i,\ \beta_i \ge 0\, \forall\, i = 1, \ldots, S\right\}.$$

Lemma 3.9 implies that we can compute $\|x-y\|$ for any pair $x,y \in \mathbb{R}^n$ using a linear program. Thus, our assumptions on the sets $\mathcal{F}(\xi^k)$ and Lemma 3.9 give rise to the following linear formulation of RecFeas($\mathcal{U}$) :

$$\begin{array}{llcll}
\min & r & & & \\
\text{s.t.} & A(\xi^k)y^k & \leq & b(\xi^k) & \text{for all } k=1,\dots,N \\
& y^k - x & = & \sum_{i=1}^S \beta_i^k e_i & \text{for all } k=1,\dots,N \\
& \sum_{i=1}^S \beta_i^k & \leq & r & \text{for all } k=1,\dots,N \\
& x & \in & \mathcal{X} & \\
& r, \beta_i^k & \geq & 0 & \text{for all } k=1,\dots,N,\ i=1,\dots,S \\
& x, y^k & \in & \mathbb{R}^n & \text{for all } k=1,\dots,N
\end{array} \tag{3.5}$$

Note that the first constraint is just the definition of the sets $\mathcal{F}(\xi^k)$. Furthermore, the second constraint together with the third constraint ensures that $\|x-y^k\| \leq r$ for all $k=1,\dots,N$. Therefore, the linear program (3.5) is equivalent to the formulation (3.3) for a finite set of scenarios each of them having a polyhedron as feasible set and if a block norm is used as distance measure. Consequently, we have shown that RecFeas($\mathcal{U}$) can be formulated as a linear program in this case. This is summarized in the following result.

Theorem 3.10. *Consider an uncertain optimization problem* (P(ξ), $\xi \in \mathcal{U}$) *with linear constraints $F(x,\xi) = A(\xi)x - b(\xi) \leq 0$ for any fixed ξ. Let $\mathcal{U} = \{\xi^k : k=1,\dots,N\}$ be a finite set and let d be induced by a block norm $\|\cdot\|$ with extreme points $e_1,\dots,e_S$. Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a polyhedron. Then* RecFeas($\mathcal{U}$) *is given as the linear program (3.5).*

If the number of constraints defining $\mathcal{X}$ and the number of extreme points of B depend at most polynomially on the dimension n, then RecFeas($\mathcal{U}$) *can be solved in polynomial time.*

We note that block norms may be generalized to the broader class of polyhedral gauges where the symmetry assumption on the unit ball is dropped (see e.g. [NP09]). Nevertheless, as it is readily shown that Lemma 3.9 applies to polyhedral gauges as well, Theorem 3.10 also holds for distance functions derived from polyhedral gauges.

Problems with hyperplanes as feasible sets. We consider another special case in which RecFeas($\mathcal{U}$) can be rewritten as a linear program, even though the distance measure does not need to be derived from a block norm: For $a \in \mathbb{R}^n, b \in \mathbb{R}$, let

$$\text{P}(a,b) \qquad \begin{array}{l} \min f(x,(a,b)) \\ \text{s.t. } F(x,(a,b)) := a^t x - b = 0 \\ x \in \mathcal{X} \end{array}$$

with $\mathcal{U} = \{(a^1,b^1),\dots,(a^N,b^N)\}$, $a^1,\dots,a^N \neq 0$. Concerning the distance measure d, we assume that it is induced by a norm $\|\cdot\|$, and we allow $\mathcal{X} \subseteq \mathbb{R}^n$ to be any polyhedral set.

RecFeas($\mathcal{U}$) is then given by

$$\begin{aligned} &\min r \\ &\text{s.t. } d(x, H_{a^k,b^k}) \le r \text{ for all } k = 1, \dots, N \\ &\phantom{\text{s.t. }} x \in \mathcal{X}, r \in \mathbb{R}, \end{aligned}$$

where $H_{a^k,b^k} = \{x \in \mathbb{R}^n : {a^k}^t x = b^k\}$ denotes the hyperplane given by (a^k, b^k). Recall the point-to-hyperplane distance [PC01]

$$d(x, H_{a,b}) = \frac{|a^t x - b|}{\|a\|^\circ},$$

where $\|\cdot\|^\circ$ denotes the dual norm to $\|\cdot\|$. As the values of $\|a^k\|^\circ$ can be precomputed and the absolute value linearized, we gain a linear program

$$\begin{aligned} &\min r \\ &\text{s.t. } -r\|a^k\|^\circ \le {a^k}^t x - b \le r\|a^k\|^\circ \text{ for all } k = 1, \dots, N \\ &\phantom{\text{s.t. }} x \in \mathcal{X}, r \in \mathbb{R}. \end{aligned} \tag{3.6}$$

Corollary 3.11. *Consider an uncertain optimization problem* $(\mathrm{P}(a,b), (a,b) \in \mathcal{U})$ *with linear constraints* $F(x,(a,b)) = a^t x - b = 0, a \in \mathbb{R}^n, b \in \mathbb{R}$ *and finite uncertainty set* $\mathcal{U} = \{(a^k, b^k) : k = 1, \dots, N\} \subseteq \mathbb{R}^{n+1}$. *Let* $\mathcal{X} \subseteq \mathbb{R}^n$ *be a polyhedral set and let* d *be induced by a norm* $\|\cdot\|$. *Then* RecFeas($\mathcal{U}$) *is given by the linear program (3.6) and can be solved in polynomial time, provided that the dual norm of* $\|\cdot\|$ *can be evaluated in polynomial time.*

Note that this result can even be extended to half-spaces as feasible solutions instead of hyperplanes. In that case, the distance would be given by

$$d(x, H^+_{a,b}) = \frac{|a^t x - b|^+}{\|a\|^\circ},$$

where $|a^t x - b|^+ = \max\{a^t x - b, 0\}$, resulting in the linear program

$$\begin{aligned} &\min r \\ &\text{s.t. } {a^k}^t x - b \le r\|a^k\|^\circ \text{ for all } k = 1, \dots, N \\ &\phantom{\text{s.t. }} r \ge 0 \\ &\phantom{\text{s.t. }} x \in \mathcal{X}, r \in \mathbb{R}. \end{aligned}$$

3.1.4 Reduction Approaches for Infinite Uncertainty Sets

In this section we consider the case of infinite uncertainty sets $\mathcal{U}$. The main goal is to reduce the set $\mathcal{U}$ to a smaller (maybe even finite) set $\mathcal{U}' \subseteq \mathcal{U}$, such that a solution of RecFeas($\mathcal{U}'$) is also a solution to RecFeas($\mathcal{U}$).

Some simple reductions can always be done: Similar to the reduction rules for set covering problems [TSRB71], we obtain the following straightforward result.

Lemma 3.12. *Consider* $\mathrm{RecFeas}(\mathcal{U})$ *for some uncertain optimization problem* $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ *with a closed set* $\mathcal{X} \subseteq \mathbb{R}^n$ *and let* d *be induced by a norm. Let* $\mathcal{F}(\xi_1) \subseteq \mathcal{F}(\xi_2)$ *for* $\xi_1, \xi_2 \in \mathcal{U}$*. Then scenario* ξ_2 *may be ignored, i.e.* $\mathrm{RecFeas}(\mathcal{U})$ *is equivalent to* $\mathrm{RecFeas}(\mathcal{U} \setminus \{\xi_2\})$.

Note that dependent on the definition of the optimization problem and the uncertainty set $\mathcal{U}$, often large classes of scenarios may be dropped. This is in particular the case if the sets $\mathcal{F}(\xi)$ are nested.

In the following we are therefore interested in identifying a kind of *core set* $\mathcal{U}' \subseteq \mathcal{U}$ containing a *finite* number of scenarios which are sufficient to consider in order to solve the recovery-robust counterpart. More precisely, we look for a finite set $\mathcal{U}'$ such that $\mathrm{RecFeas}(\mathcal{U}')$ and $\mathrm{RecFeas}(\mathcal{U})$ are equivalent.

In the following we consider a polytope $\mathcal{U}$ with a finite number of extreme points $\xi^1, \dots, \xi^N$, i.e. let

$$\mathcal{U} = \mathrm{conv}(\mathcal{U}') \text{ where } \mathcal{U}' = \{\xi^1, \dots, \xi^N\}.$$

Then many robustness concepts have (under mild conditions) the following property: Instead of investigating *all* $\xi \in \mathcal{U}$, it is enough to investigate the extreme points $\xi^1, \dots, \xi^N$ of $\mathcal{U}$. This leads to formulations of the robust counterpart which are of the same type as the original problem. For example, for the strictly robust counterpart $\mathrm{SR}(\mathcal{U})$ of an uncertain optimization problem $(P(\xi), \xi \in \mathcal{U} = \mathrm{conv}\{\xi^1, \dots, \xi^N\})$ it is known that

$$F(x, \xi) \le 0 \text{ for all } \xi \in \mathcal{U} \text{ if and only if } F(x, \xi^k) \le 0 \text{ for all } k = 1, \dots, N,$$

if $F(x, \cdot)$ is quasiconvex for all $x \in \mathcal{X}$. Hence, under such conditions, it follows that $\mathrm{SR}(\mathcal{U})$ is equivalent to $\mathrm{SR}(\{\xi^1, \dots, \xi^N\})$ if the uncertainty in the constraints and in the objective is quasiconvex. In particular, if $\mathrm{P}(\xi)$ is a linear program and $\mathcal{U}$ a polyhedral set, also the strictly robust counterpart $\mathrm{RC}(\mathcal{U})$ is a linear program.

Unfortunately, a similar result for the recovery-to-feasibility counterpart does not hold. This means that the solution of $\mathrm{RecFeas}(\mathcal{U}')$ for the set $\mathcal{U}'$ does in general not help for identifying a solution of $\mathrm{RecFeas}(\mathcal{U})$ with respect to the larger set $\mathcal{U} = \mathrm{conv}(\mathcal{U}')$, as demonstrated in the following example.

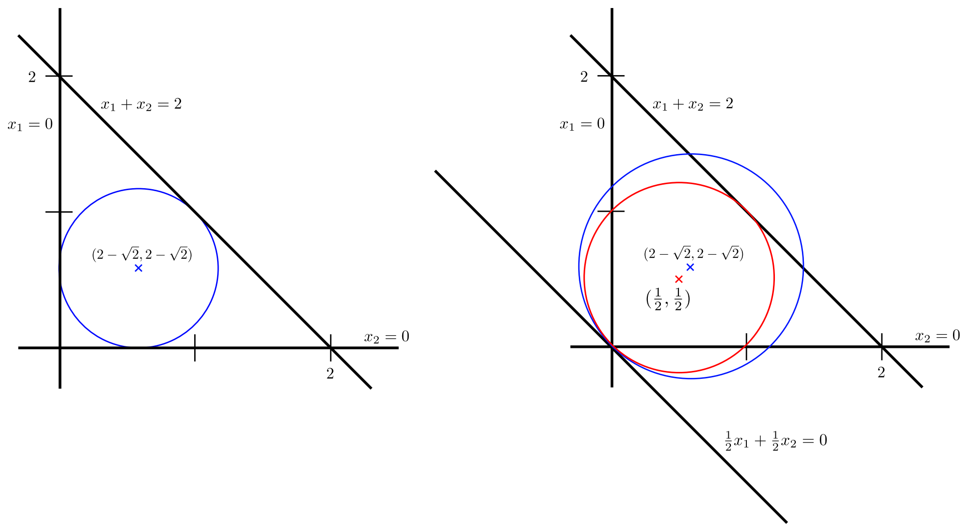
Example 3.13. *Consider the following uncertain optimization problem:*

$$\begin{array}{lll} P(a_1, a_2, b) & \min & f(x_1, x_2) \\ & s.t. & a_1 x_1 + a_2 x_2 - b = 0 \\ & & x_1, x_2 \in \mathbb{R}, \end{array}$$

where

$$\mathcal{U} = \mathrm{conv}\left\{(1,0,0), (0,1,0), (1,1,2)\right\}.$$

Let the recovery distance be the Euclidean distance. Then the optimal solution with respect to RecFeas$(\mathcal{U}')$*, where* $\mathcal{U}' = \{(1,0,0), (0,1,0), (1,1,2)\}$*, is given by* $x^* = (2-$


2
$x_1 = 0$
$x_1 + x_2 = 2$
$(2-\sqrt{2}, 2-\sqrt{2})$
$x_2 = 0$
2
2
$x_1 = 0$
$x_1 + x_2 = 2$
$(2-\sqrt{2}, 2-\sqrt{2})$
$(\frac{1}{2}, \frac{1}{2})$
$x_2 = 0$
2
$\frac{1}{2}x_1 + \frac{1}{2}x_2 = 0$

(a) Optimal solution w.r.t. the extreme points of $\mathcal{U}$.

(b) Optimal solution w.r.t. the whole set $\mathcal{U}$.

Figure 3.2 RecFeas($\mathcal{U}'$) and RecFeas($\mathcal{U}$) may have different optimal solutions.

$\sqrt{2}, 2-\sqrt{2})$, *the midpoint of the incircle of the triangle that is given by the intersections of the respective feasible solutions, see Figure 3.2(a).*

On the other hand, this solution is not optimal anymore when the convex hull of $\mathcal{U}'$ is taken into consideration. Indeed, by elementary geometry, one finds that

$$r(x^*, \mathcal{U}) = \sqrt{2} \cdot (2-\sqrt{2}) \approx 0.828,$$
$$r(\bar{x}, \mathcal{U}) = \frac{1}{\sqrt{2}} \approx 0.707,$$

where $\bar{x} = (\frac{1}{2}, \frac{1}{2})$. Therefore, solving RecFeas($\mathcal{U}'$) does not give an optimal solution to RecFeas($\mathcal{U}$).

However, in the following we show that the center with respect to the sets $\mathcal{F}(\xi^1), \ldots, \mathcal{F}(\xi^N)$ can at least be used as an approximation of the optimal solution to RecFeas($\mathcal{U}$) in the sense that its reliability (as defined in [BTN00], see also Section 2.7) is not too bad, and derive cases in which RecFeas(conv$\{\xi^1, \ldots, \xi^N\}$) is in fact equivalent to RecFeas($\{\xi^1, \ldots, \xi^N\}$). Let us denote

$$\begin{array}{rcl} \mathcal{U}' & = & \{\xi^1, \ldots, \xi^N\}, \\ x' & & \text{as the center w.r.t. } \{\mathcal{F}(\xi) : \xi \in \mathcal{U}'\} \text{ and} \\ z' & = & r(x', \mathcal{U}') \text{ be its radius w.r.t. } \mathcal{U}'. \end{array}$$

We start by showing that an optimal solution to RecFeas($\mathcal{U}'$) does not violate the constraints $F(\cdot, \xi), \xi \in \mathcal{U}$ too much.

Theorem 3.14. *Let* $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ *be an uncertain optimization problem. We assume that* $F_i(x,\cdot)$ *is quasiconvex for all* $i = 1,\dots,m$ *and all fixed* x*. Furthermore, let* $g_i^k(x) := F_i(x,\xi^k)$ *be differentiable with bounded gradients* $\|\nabla g_i^k(x)\|_\infty \le \eta$ *for all* $x \in \mathbb{R}^n$*, for all* $k = 1,\dots,N$*.*

We consider a closed set $\mathcal{X} \subseteq \mathbb{R}^n$ *and a polyhedral uncertainty set* $\mathcal{U} = \mathrm{conv}(\mathcal{U}')$*,* $\mathcal{U}' = \{\xi^1,\dots,\xi^N\}$*. Let* d *be induced by a norm* $\|\cdot\|$ *and let* c *be such that* $\|x\|_1 \le c\|x\|$ *for all* $x \in \mathbb{R}^n$*.*

Let x' *be an optimal solution to* $\mathrm{RecFeas}(\mathcal{U}')$ *and* $z' = r(x',\mathcal{U}')$ *be the corresponding radius. Then*

$$F_i(x',\xi) \le c\cdot\eta\cdot z' \quad \forall \xi \in \mathcal{U},\ \forall i = 1,\dots,m,$$

i.e. x' *is reliable for* $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ *with respect to* $c\eta z'$*.*

Proof. As x' is a center with radius z' there exist x^k, $k = 1,\dots,N$ with $F_i(x^k,\xi^k) = g_i^k(x^k) \le 0$ for $i = 1,\dots,m$ and $\|x^k - x'\| \le z'$.

Fix k and i. Then the mean value theorem (or, alternatively, the Taylor expansion of g_i^k at x^k) gives

$$g_i^k(x') = g_i^k(x^k) + \nabla g_i^k(\zeta)\cdot(x' - x^k)$$

for a suitable $\zeta \in \mathbb{R}^n$. We estimate

$$\begin{aligned}
g_i^k(x') &= \underbrace{g_i^k(x^k)}_{\le 0} + \nabla g_i^k(\zeta)\cdot(x' - x^k)\\
&\le |\nabla g_i^k(\zeta)\cdot(x' - x^k)|\\
&\le \|\nabla g_i^k(\zeta)\|_\infty \cdot \|x' - x^k\|_1\\
&\le \eta c\|x' - x^k\|\\
&\le \eta c z'
\end{aligned}$$

Let $\xi \in \mathcal{U}$ be arbitrary, then there exists $\lambda \in \mathbb{R}^N_+$ such that $\xi = \sum_{k=1}^N \lambda_k \xi^k$ and $\sum_{k=1}^N \lambda_k = 1$. Due to the quasiconvex uncertainty in the constraints we conclude for every component $i = 1,\dots,m$ that

$$\begin{aligned}
F_i(x',\xi) &= F_i(x', \sum_{k=1}^N \lambda_k \xi^k)\\
&\le \max_{k=1}^N F_i(x',\xi^k) = \max_{k=1}^N g_i^k(x')\\
&\le \eta c z'
\end{aligned}$$

which completes the proof. □

It would be desirable to bound not only the deviation in the constraints but also the distance from the center x' to any set $\mathcal{F}(\xi)$ for arbitrary $\xi \in \mathcal{U}$. Unfortunately,

$$d(x',\mathcal{F}(\xi)) \le z'$$

does not necessarily hold if $\xi \notin \mathcal{U}'$ as Example 3.13 demonstrates. However, for problems with specific structure, more can be said.

Theorem 3.15. *Let* $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ *be an uncertain optimization problem with uncertainty set* $\mathcal{U} = \mathrm{conv}(\mathcal{U}')$ *with* $\mathcal{U}' := \{\xi^1, \dots, \xi^N\}$*. Let* $F : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}$ *be jointly quasiconvex in the arguments* (x, ξ)*. Let* d *be induced by a norm* $\|\cdot\|$*. Then, for any* $x \in \mathbb{R}^n$*, the function* $\mathcal{U} \ni \xi \longmapsto d(x, \mathcal{F}(\xi))$ *is quasiconvex. In particular, for all* $x \in \mathbb{R}^n$

$$\max_{\xi \in \mathrm{conv}(\mathcal{U}')} d(x, \mathcal{F}(\xi)) = \max_{\xi \in \mathcal{U}'} d(x, \mathcal{F}(\xi)),$$

i.e. for any closed set $\mathcal{X} \subseteq \mathbb{R}^n$ *we have that* $\mathrm{RecFeas}(\mathcal{U})$ *and* $\mathrm{RecFeas}(\mathcal{U}')$ *are equivalent.*

Proof. Let $x \in \mathbb{R}^n$ and $\xi^1, \xi^2 \in \mathcal{U}$ be given. Set $\alpha := \max_{i=1,2}\{d(x, \mathcal{F}(\xi^i))\}$. We need to show that, for any λ, $0 \le \lambda \le 1$, we also have

$$d(x, \mathcal{F}((1-\lambda)\xi^1 + \lambda\xi^2)) \le \alpha.$$

For $i = 1, 2$, let $y^i \in \mathcal{F}(\xi^i)$ such that $\|y^i - x\| \le \alpha$. Since $y^i \in \mathcal{F}(\xi^i)$, we have by the quasiconvexity of F that

$$F((1-\lambda)(y^1, \xi^1) + \lambda(y^2, \xi^2)) \le \max_{i=1,2} \left\{F(y^i, \xi^i)\right\} \le 0.$$

Hence,

$$(1-\lambda)y^1 + \lambda y^2 \in \mathcal{F}((1-\lambda)\xi^1 + \lambda\xi^2),$$

and

$$\begin{aligned} \|(1-\lambda)y^1 + \lambda y^2 - x\| &= \|(1-\lambda)(y^1 - x) + \lambda(y^2 - x)\| \\ &\le \|(1-\lambda)(y^1 - x)\| + \|\lambda(y^2 - x)\| \le \alpha. \end{aligned}$$

We finally obtain that

$$d(x, \mathcal{F}((1-\lambda)\xi^1 + \lambda\xi^2)) \le \|(1-\lambda)y^1 + \lambda y^2 - x\| \le \alpha,$$

i.e. $\mathcal{U} \ni \xi \longmapsto d(x, \mathcal{F}(\xi))$ is quasiconvex.

Using this result we can conclude the second assertion: First, it is clear that

$$\max_{\xi \in \mathrm{conv}(\mathcal{U}')} d(x, \mathcal{F}(\xi)) \ge \max_{\xi \in \mathcal{U}'} d(x, \mathcal{F}(\xi)).$$

To see the other inequality, take some $\xi \in \mathrm{conv}(\mathcal{U}')$, i.e. there exist $\lambda_k, k = 1, \dots, N$ with $0 \le \lambda_k \le 1$ for all $k = 1, \dots, N$ and $\sum_{k=1}^N \lambda_k = 1$ such that $\xi = \sum_{k=1}^N \lambda_k \xi^k$. Due to the quasiconvexity of $d(x, \mathcal{F}(\cdot))$, we obtain

$$d(x, \mathcal{F}(\xi)) = d(x, \mathcal{F}(\sum_{k=1}^N \lambda_k \xi^k)) \le \max_{k=1,\dots,N} d(x, \mathcal{F}(\xi^k)).$$

This is true for all $\xi \in \operatorname{conv}(\mathcal{U}')$, thus we conclude

$$\max_{\xi\in\operatorname{conv}(\mathcal{U}')} d(x,\mathcal{F}(\xi)) \leq \max_{k=1,\ldots,N} d(x,\mathcal{F}(\xi^k)).$$

□

An important particular case of application of Theorem 3.15 is the case in which

$$F(x,\xi) = G(x) - b(\xi)$$

for a convex G and concave b (i.e. the uncertainty is in the right-hand side), since F is then jointly quasiconvex in (x,ξ).

Corollary 3.16. *Let $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ be an uncertain optimization problem with uncertainty set $\mathcal{U} = \operatorname{conv}(\mathcal{U}')$ with $\mathcal{U}' := \{\xi^1,\ldots,\xi^N\}$. Let $\mathcal{X} \subseteq \mathbb{R}^n$ be closed, d be induced by a norm and let $F(x,\xi) = G(x) - b(\xi)$ with a convex function $G : \mathbb{R}^n \to \mathbb{R}$ and a concave function $b(\xi) : \mathbb{R}^N \to \mathbb{R}^m$. Then* RecFeas($\mathcal{U}$) *and* RecFeas($\mathcal{U}'$) *are equivalent.*

We remark that G must not depend on the scenario ξ. Example 3.13 shows that Corollary 3.16 is not even true for a linear function $F(x,\xi) = A(\xi)x - b(\xi)$: If the matrix A is dependent on ξ, we cannot conclude that RecFeas($\mathcal{U}$) and RecFeas($\mathcal{U}'$) are equivalent.

Note that Corollary 3.16 applies in particular for the special case that $b(\xi) = b$, i.e. for uncertain convex optimization problems of the type

$$\mathrm{P}(b) \quad \min_{x\in\mathbb{R}^n}\{f(x) : G(x) \leq b\}. \tag{3.7}$$

In particular we know for $\mathrm{P}(b)$ that the center with respect to some finite set $\mathcal{U}'$ solves the uncertain problem with respect to $\mathcal{U} = \operatorname{conv}(\mathcal{U}')$.

This means we can use the finite set $\mathcal{U}'$ instead of $\mathcal{U}$ when solving RecFeas($\mathcal{U}$) if the conditions of the previous theorem apply. This is summarized next.

Corollary 3.17. *Let $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ be an uncertain optimization problem with uncertainty set $\mathcal{U} = \operatorname{conv}(\mathcal{U}')$ with $\mathcal{U}' := \{\xi^1,\ldots,\xi^N\}$ and with constraints $F(x,\xi) = G(x) - b(\xi)$ with a convex function $G : \mathbb{R}^n \to \mathbb{R}$ and a concave function $b(\xi) : \mathbb{R}^N \to \mathbb{R}^m$. Let $\mathcal{X} \subseteq \mathbb{R}^n$ be convex and closed and d be induced by a norm $\|\cdot\|$. Then* RecFeas($\mathcal{U}$) *can be formulated as the following convex program:*

$$\begin{array}{lllll}
\min & r & & & \\
s.t. & G(y^k) & \leq & b(\xi^k) & \text{for all } k = 1,\ldots,N \\
 & \|y^k - x\| & \leq & r & \text{for all } k = 1,\ldots,N \\
 & x & \in & \mathcal{X} & \\
 & r & \in & \mathbb{R} & \\
 & x, y^k & \in & \mathbb{R}^n & \text{for all } k = 1,\ldots,N
\end{array} \tag{3.8}$$

Combining this corollary with Theorem 3.10 from Section 3.1.3, we obtain the following result: The recovery-robust counterpart of an optimization problem with convex uncertainty which is only in its right-hand side and with polyhedral uncertainty set can be formulated as a linear program if a block norm is used to measure the recovery costs. In particular, the recovery-robust counterpart of such a linear program under polyhedral uncertainty sets and block norms as distance functions remains a linear program.

Theorem 3.18. *Let* $(\mathrm{P}(\xi), \xi \in \mathcal{U})$ *be an uncertain optimization problem with uncertainty set* $\mathcal{U} = \mathrm{conv}(\mathcal{U}')$ *with* $\mathcal{U}' := \{\xi^1, \dots, \xi^N\}$ *and with constraints defined by a polyhedron* $\mathcal{X} \subseteq \mathbb{R}^n$*, linear functions* $F(x, \xi) = A(x) - b(\xi)$ *for any fixed* $\xi \in \mathcal{U}$ *and a concave function* $b(\xi) : \mathbb{R}^N \to \mathbb{R}^m$*. Furthermore, let* $\| \cdot \|$ *be a block norm.*

Then $\mathrm{RecFeas}(\mathcal{U})$ *can be formulated as linear program. If the terms defining* $\mathcal{X}$ *and the number of extreme points of the block norm depend at most polynomially on the dimension* n*, then the problem can be solved in polynomial time.*

Proof. According to Theorem 3.16 we can replace $\mathcal{U}$ by the finite set $\mathcal{U}'$ in the recovery-robust counterpart, i.e. we consider $\mathrm{RecFeas}(\mathcal{U}')$ instead of $\mathrm{RecFeas}(\mathcal{U})$. We are hence left with a problem for which the assumptions of Theorem 3.10 are satisfied yielding a formulation as linear program. □

Note that many practical applications satisfy the conditions of Theorem 3.18. Among these are scheduling– and timetabling problems, where the uncertainty is the length of the single tasks to be completed, and hence in the common linear formulations in the right-hand side (see Section 4.2).

3.1.5 Algorithms for RecFeas

We now consider two algorithms to solve RecFeas: In Section 3.1.5.1, an iterative approach for finite uncertainty sets is presented that alternately solves a location problem for the robust solution, and a location problem for each of the scenarios, and a sampling approach for infinite scenarios in Section 3.1.5.2.

3.1.5.1 An Iterative Approach

We consider an uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ with a finite uncertainty set $\mathcal{U} = \{\xi^1, \dots, \xi^N\}$. A typical RecFeas robust counterpart would then have a constraint structure as depicted in Figure 3.3: For every scenario, a copy of the problem needs to be solved, and constraints are added that model the distance function being used. Every row corresponds to a constraint, and every column to a variable.

This problem structure is well-suited for Dantzig-Wolfe decomposition approaches, which is a promising aspect for further research. In this section, we focus on the related approach of alternately solving the master (location) problem, and each scenario problem. Algorithm 1 describes this procedure in detail. Note that the presented

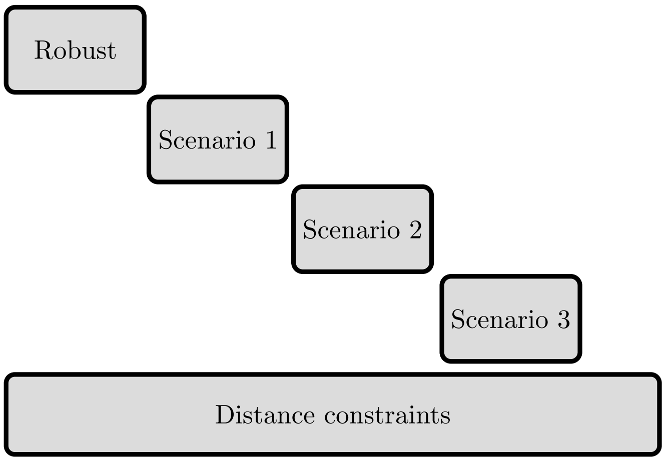


Figure 3.3 Structure of a RecFeas problem with three scenarios.

version focuses on the center problem; for the median problem, we need to minimize the sum of distances instead of the maximum distance in Step 5.

We explain Algorithm 1 using an example.

Example 3.19. *Assume that there are three scenarios $\mathcal{U} = \{\xi^1, \xi^2, \xi^3\}$ given, with the following feasible sets:*

$$\mathcal{F}(\xi^1) = [1,2] \times [1,2], \quad \mathcal{F}(\xi^2) = [1,2] \times [3,4], \quad \mathcal{F}(\xi^3) = [3,4] \times [1,2].$$

The recovery distance d is the Euclidean distance, and the feasible set for the nominal problem contains $[0,4] \times [0,4]$. We would like to find the robust solution with the smallest maximum recovery distance to a feasible solution in every scenario.

The behavior of Algorithm 1 is shown in Figure 3.4. For the starting solution $x^{(0)} = (0,0)$ we calculate the closest point $y^{(0),i}$ from the set $\mathcal{F}(\xi^i)$. We get $y^{(0),1} = (1,1)$, $y^{(0),2} = (1,3)$ and $y^{(0),3} = (3,1)$. We then solve a location problem with respect to these three points to gain $x^{(1)} = (2,2)$ as a center solution with $r^{(0)} = \sqrt{2}$. The first iteration of the algorithm ends, and we proceed to the second iteration.

We calculate the closest points $y^{(1),1} = (2,2)$, $y^{(1),2} = (2,3)$ and $y^{(1),3} = (3,2)$, and get a new center $x^{(2)} = (2 + \frac{1}{2}\sqrt{2}, 2 + \frac{1}{2}\sqrt{2})$ with radius $r^{(1)} = \frac{1}{2}\sqrt{2}$. The next iteration will calculate the same values for y and for x again, and terminate: In this case, we have found the optimal solution to RecFeas.

As a termination criterion, we considered the difference in the calculated robust solutions per iteration. We show that this is well-defined, i.e., the algorithm converges for any choice of ϵ.

Lemma 3.20. *Let the requirements of Algorithm 1 be fulfilled. Then $r^{(k)}$, $k = 0, 1, \ldots$, is a monotonically decreasing, converging sequence (for both the median and the center case).*

Algorithm 1 (Iterative RecFeas)

```
Require: An uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ with a finite uncertainty
    set $\mathcal{U} = \{\xi^1, \dots, \xi^N\}$, a distance measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and a tolerance $\epsilon > 0$.
 1: Set $x^{(0)} = 0$ and $k = 0$.
 2: for $i = 1, \dots, N$ do
 3:     For fixed $x^{(k)}$, solve
            $\min d(x^{(k)}, y^{(k),i})$
            $\text{s.t.} \quad y^{(k),i} \in \mathcal{F}(\xi^i)$
        Let $y^{(k),i}$ be a resulting optimal solution.
 4: end for
 5: For fixed $y^{(k),i}$, $i = 1, \dots, N$, solve
            $\min \max_{i=1,\dots,N} d(x^{(k+1)}, y^{(k),i})$
            $\text{s.t.} \quad x^{(k+1)} \in \mathcal{F}(\hat{\xi})$
    Set $x^{(k+1)}$ to be an optimal solution.
 6: Set $r^{(k)} = \max_{i=1,\dots,N} d(x^{(k+1)}, y^{(k),i})$.
 7: if $k > 0$ and $|r^{(k)} - r^{(k-1)}| \le \epsilon$ then
 8:     return $x^{(k+1)}$
 9: else
10:     Set $k := k + 1$.
11:     Go to Step 2.
12: end if
```

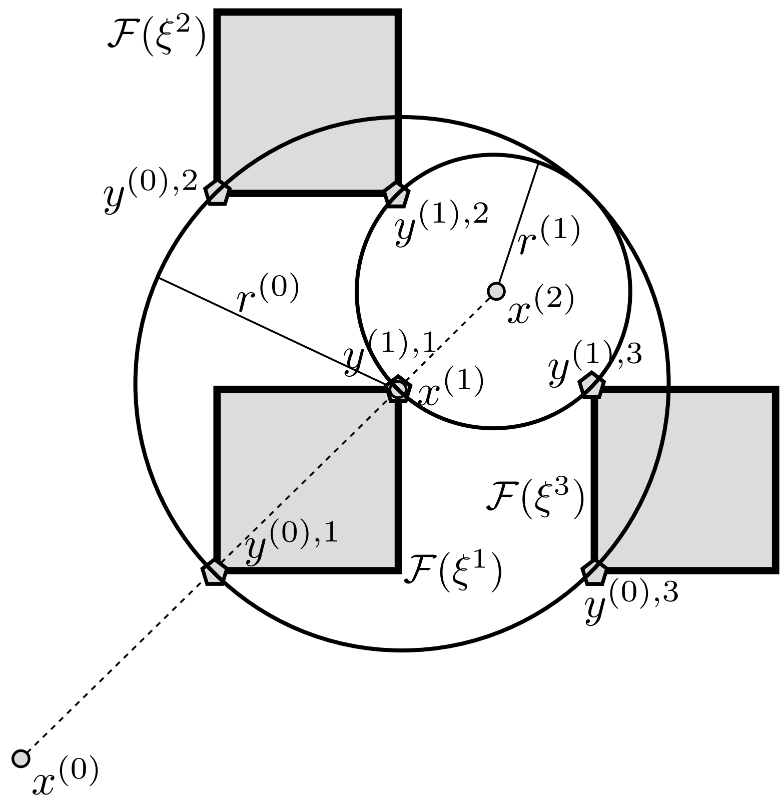


Figure 3.4 Example run of Algorithm 1.

Proof. In the center case, we have

$$x^{(k+1)} \in \arg\min_x \left\{ \max_{i=1,\ldots,N} d(x, y^{(k),i}) : x \in \mathcal{F}(\hat{\xi}) \right\},$$

and therefore,

$$r^{(k)} = \max_{i=1,\ldots,N} d(x^{(k+1)}, y^{(k),i}) \leq \max_{i=1,\ldots,N} d(x^{(k)}, y^{(k),i}).$$

As we also have

$$y^{(k),i} \in \arg\min_y \left\{ d(x^{(k)}, y) : y \in \mathcal{F}(\xi^i) \right\},$$

we get

$$d(x^{(k)}, y^{(k),i}) \leq d(x^{(k)}, y^{(k-1),i}) \quad \forall i = 1, \ldots, N$$

and therefore

$$\max_{i=1,\ldots,N} d(x^{(k)}, y^{(k),i}) \leq \max_{i=1,\ldots,N} d(x^{(k)}, y^{(k-1),i}) = r^{(k-1)},$$

which yields

$$r^{(k)} \leq r^{(k-1)},$$

i.e., $r^{(k)}$ is monotonically decreasing. As it is also bounded from below by 0, it converges. In the case of a median problem, this proof works analogously. □

Corollary 3.21. *Algorithm 1 terminates for every input.*

Even though Lemma 3.20 shows that the objective values $r^{(k)}$ converge, the solutions calculated in every iteration do not necessarily converge, even in the case of convex sets $\mathcal{F}(\xi^i)$, $i = 1, \ldots, N$ and a convex distance measure d. A counterexample is already presented in Example 3.2, where two closed, convex feasible sets are arbitrary close to each other, but do not meet. The sequence of solutions is shown in Figure 3.5.

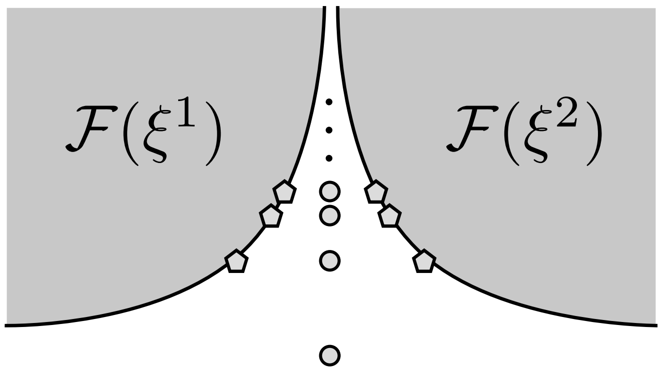


Figure 3.5 Example for which the solutions of Algorithm 1 do not converge.

However, we can conclude that the distance between the solutions converges.

Lemma 3.22. *Let the requirements of Algorithm 1 be fulfilled. Then the sequence $d(x^{(k+1)}, x^{(k)})$, $k = 0, 1, \ldots$ is bounded (for both the median and the center case).*

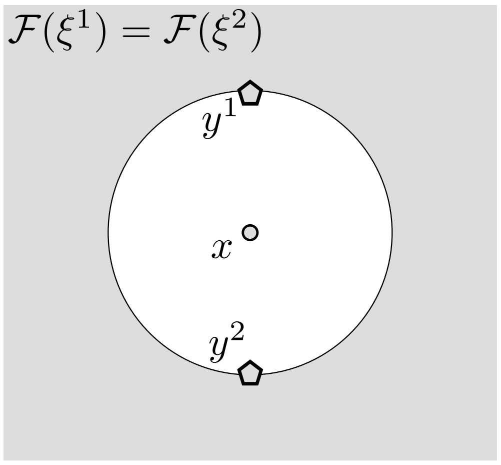
Proof. We know from Lemma 3.20 that the sequence $r^{(k)}$ converges and is monotonically decreasing. We have

$$\begin{aligned}
d(x^{(k+1)}, x^{(k)}) &\leq d(x^{(k+1)}, y^{(k),1}) + d(y^{(k),1}, x^{(k)}) \\
&\leq \max_{i=1,\ldots,N} d(x^{(k+1)}, y^{(k),i}) + \max_{i=1,\ldots,N} d(x^{(k)}, y^{(k),i}) \\
&\leq \max_{i=1,\ldots,N} d(x^{(k+1)}, y^{(k),i}) + \max_{i=1,\ldots,N} d(x^{(k)}, y^{(k-1),i}) \\
&= r^{(k)} + r^{(k-1)} \\
&\leq 2r^{(k-1)}
\end{aligned}$$

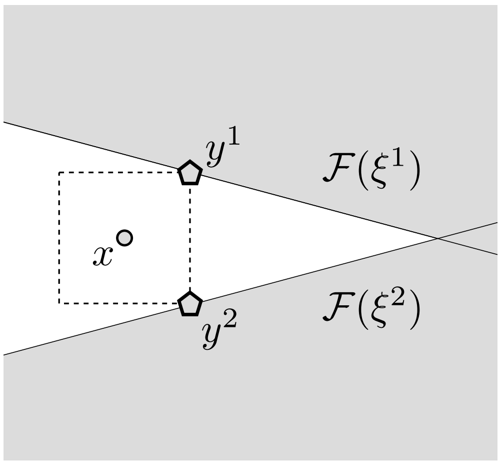
and therefore, $d(x^{(k+1)}, x^{(k)})$ is bounded. The proof for the median case works analogously. □

However, even though $d(x^{(k+1)}, x^{(k)})$ is bounded, it does not necessarily converge: As an example, consider the space $\mathbb{R}^2$ with the two scenarios $\mathcal{F}(\xi^1) = \{(x_1, x_2) : x_2 \geq 1\}$ and $\mathcal{F}(\xi^2) = \{(x_1, x_2) : x_2 \leq -1\}$. We use the l_∞ distance. Beginning with $x^0 = (0, 0)$, we immediately have an optimal radius of 1, but may have a distance from $x^{(k)}$ to $x^{(k+1)}$ of up to 2.

Furthermore, convergence of $r^{(k)}$ and of $d(x^{(k+1)}, x^{(k)})$ neither shows the convergence of $x^{(k)}$, nor that $x^{(k)}$ converges to a local optimum, if it converges at all. We present two examples in Figure 3.6.



(a) Non-convex feasible sets $\mathcal{F}(\xi)$.



(b) Non-Euclidean recovery distance.

Figure 3.6 Two examples where Algorithm 1 converges, but solutions are not optimal.

In the first case, two scenarios ξ^1, ξ^2 are given with equal feasible sets $\mathcal{F}(\xi^1) = \mathcal{F}(\xi^2)$. These sets are not convex, as there is a spherical "hole" of infeasible solutions. We would like to find a center with respect to the l_2 distance. Algorithm 1 terminates for the values of x, y^1 and y^2 as shown in Figure 3.6(a): x minimizes the distance to y^1 and y^2, while y^1 and y^2 both minimize the distance to x. However, x is a local maximum, and the best objective value for (RecFeas) is 0, as $\mathbb{SR} = \mathcal{F}(\xi^1) = \mathcal{F}(\xi^2)$.

In the second case, we consider the l_∞ norm and would like to minimize the maximum distance to two intersecting halfspaces $\mathcal{F}(\xi^1)$, $\mathcal{F}(\xi^2)$. Again, $\mathbb{SR} \neq \emptyset$ and (RecFeas) has an optimal solution with objective value 0. However, Algorithm 1 converges for the points x, y^1 and y^2 as shown in Figure 3.6(b). x minimizes the distance to y^1 and y^2, while y^1 and y^2 both minimize the distance to x.

We now analyze circumstances where such cases are impossible.

Lemma 3.23. *Let the requirements of Algorithm 1 be fulfilled. Let $d = l_2$ be the Euclidean distance, and $\mathcal{F}(\xi)$ be convex and closed for all $\xi \in \mathcal{U}$. Let the algorithm terminate after k steps with $(x^*, y^{1*}, \ldots, y^{N*}) := (x^{(k+1)}, y^{(k),1}, \ldots, y^{(k),N}) = (x^{(k)}, y^{(k-1),1}, \ldots, y^{(k-1),N})$. Set*

$$H_i := \{y^{i*} + z : z^t(y^{i*} - x^*) = 0, z \in \mathbb{R}^n\}$$

and define $d_H(x) = \min_{y \in H} d(x, y)$. Then x^* is minimizer of $\max_{i \in 1,\ldots,N} d_{H_i}(\cdot)$.

Proof. We need to show that $\max_{i=1,\ldots,N} d_{H_i}(x) \geq \max_{i=1,\ldots,N} d_{H_i}(x^*)$ for all $x \in \mathbb{R}^n$. Note that

$$d_{H_i}(x) = \frac{|(y^{i*} - x^*)^t x - (y^{i*} - x^*)^t y^{i*}|}{\|y^{i*} - x^*\|_2}$$

We have

$$\begin{aligned}
\max_{i=1,\ldots,N} d_{H_i}(x) &= \max_{i=1,\ldots,N} \frac{|(y^{i*} - x^*)^t x - (y^{i*} - x^*)^t y^{i*}|}{\|y^{i*} - x^*\|_2} \\
&= \max_{i=1,\ldots,N} \frac{|\left\langle x - y^{i*}, x^* - y^{i*} \right\rangle|}{\|y^{i*} - x^*\|_2} \\
&= \max_{i=1,\ldots,N} \frac{|\left\langle x^* - y^{i*}, x^* - y^{i*} \right\rangle + \left\langle x - x^*, x^* - y^{i*} \right\rangle|}{\|y^{i*} - x^*\|_2} \\
&= \max_{i=1,\ldots,N} \frac{|\|y^{i*} - x^*\|_2^2 + \left\langle x - x^*, x^* - y^{i*} \right\rangle|}{\|y^{i*} - x^*\|_2} \\
&= \max_{i=1,\ldots,N} \frac{|\|y^{i*} - x^*\|_2^2 + \left\langle z, x^* - y^{i*} \right\rangle|}{\|y^{i*} - x^*\|_2}
\end{aligned}$$

where $x = x^* + z$. Now, assume that z is such that

$$\left\langle z, x^* - y^{i*} \right\rangle < 0$$

for all i with $d(x^*, y^{i*}) = r(x^*)$, i.e. there is a separating hyperplane between x^* and the critical points y^{i*}. Therefore, for λ small enough we have that

$$\max_{i=1,\ldots,N} d(x^* + \lambda z, y^{i*}) < \max_{i=1,\ldots,N} d(x^*, y^{i*})$$

which contradicts the assumption that Algorithm 1 terminated.

We conclude that that there is no z such that $\left\langle z, x^* - y^{i*} \right\rangle < 0$ for all i with $d(x^*, y^{i*}) = r(x^*)$, and therefore

$$\max_{i=1,\ldots,N} d_{H_i}(x) \geq \max_{i=1,\ldots,N} \frac{\|y^{i*} - x^*\|_2^2}{\|y^{i*} - x^*\|_2} = \max_{i=1,\ldots,N} d_{H_i}(x^*)$$

□

Theorem 3.24. *Let the requirements of Algorithm 1 be fulfilled. Let $d = l_2$ be the Euclidean distance, and $\mathcal{F}(\xi)$ be convex and closed for all $\xi \in \mathcal{U}$. Let the algorithm terminate after k steps with $(x^*, y^{1*}, \dots, y^{N*}) := (x^{(k+1)}, y^{(k),1}, \dots, y^{(k),N}) = (x^{(k)}, y^{(k-1),1}, \dots, y^{(k-1),N})$. Then x^* is a global optimum for RecFeas in the center case.*

Proof. We reconsider the tangential hyperplanes H_i at the feasible points $y^{i*} \neq x^*$:

$$H_i := \{y^{i*} + z : z^t(y^{i*} - x^*) = 0, z \in \mathbb{R}^n\}$$

Again, denote by $d_H(x)$ the distance of point x to the hyperplane H_i, i.e. $d_H(x) = \min_{y \in H} d(x, y)$. Because $\mathcal{F}(\xi^i)$ is convex, we have $d_{H_i}(x) \leq d(x, \mathcal{F}(\xi^i)) \leq r(x)$ for all x from the same halfspace as x^*. Furthermore, $r(x^*) = \max_{i=1,\dots,N} d_{H_i}(x^*)$, and x^* is minimizer of $\max_{i \in 1,\dots,N} d_{H_i}(\cdot)$ due to Lemma 3.23.

We consider a neighborhood $V \subseteq \mathbb{R}^n$ of x^* that lies in the same halfspace as x^* for all H_i. Assume that there is $x' \in V$ such that $r(x') \leq r(x^*)$. Then

$$r(x') \leq r(x^*) = \max_{i=1,\dots,N} d_{H_i}(x^*) \leq \max_{i=1,\dots,N} d_{H_i}(x') \leq r(x')$$

and therefore $r(x') = r(x^*)$, i.e., x^* is a local optimum. Due to Lemma 3.6, x^* is also a global optimum of RecFeas.

For points $y^{i*} = x^*$, we get halfspaces that contain x^*, and the corresponding scenarios are not in $WC(x^*,\mathcal{U})$ if $r(x^*) > 0$. The case $r(x^*) = 0$ is trivial. □

Theorem 3.25. *We consider the center case. Let the requirements of Algorithm 1 be fulfilled. Let $d = l_2$ be the Euclidean distance, and let $\mathcal{F}(\xi^i)$ be convex and closed for all $i \in \{1, \dots, N\}$. We assume that a unique optimizer of (RecFeas) exists, and that $d(x^{(k+1)}, x^{(k)})$ converges. Then the sequence $r^{(k)}$, $k = 0, 1, \dots$, converges to the optimal value of RecFeas, and $x^{(k)}$, $k = 0, 1, \dots$, converges to the optimal solution of RecFeas.*

Proof. Let x^* be the optimal solution with radius r^*.

We distinguish the following cases:

1. $r(x^{(k)}) \to r' > r^*$.

   We claim that for all $\epsilon > 0$ there are K and $\delta > 0$ such that $r(x^{(k)}) \leq r(x) + \epsilon$ for all $k > K$ and x with $d(x, x^{(k)}) \leq \delta$. To see this, first note that $d(y^{(k+1),i}, y^{(k),i}) \to 0$ as $\mathcal{F}(\xi^i)$ is convex and d is strictly convex, and $d(x^{(k+1)}, x^{(k)}) \to 0$. Set K such that $d(x^{(k+1)}, x^{(k)}) \leq \epsilon/2$ and $d(y^{(k+1)}, y^{(k)}) \leq \epsilon/2$. Set $\delta = \epsilon/2$. For $k > K$, let x with $d(x, x^{(k)}) \leq \delta$ be given, and let $y^i = \arg\min\{d(x, y) : y \in \mathcal{F}(\xi^i)\}$. Then

$$\begin{aligned} d(x, y^i) &\leq d(x, y^{(k),i}) + d(y^{(k),i}, y^i) \\ &\leq d(x^{(k)}, y^{(k),i}) + d(x, x^{(k)}) + d(y^{(k),i}, y^i) \\ &\leq r(x^{(k)}) + \delta/2 + \delta/2 \end{aligned}$$

$$\leq r(x^{(k)}) + \epsilon$$

This proves the claim. Due to convexity of r and $\mathcal{F}(\xi^i)$, we get for ϵ and λ small enough:

$$\begin{aligned} r(x^{(k)}) &\leq r(\lambda x^{(k)} + (1-\lambda)x^*) + \epsilon \\ &\leq \lambda r(x^{(k)}) + (1-\lambda) r(x^*) + \epsilon \end{aligned}$$

and therefore

$$r(x^{(k)}) \leq r(x^*) + \frac{\epsilon}{1-\lambda}$$

This contradicts $r(x^{(k)}) \to r' > r^*$.

2. $r(x^{(k)}) \to r' = r^*$ and $x^{(k)}$ does not converge.

   As x^* is a unique optimal solution and r is continuous and convex, for all $\epsilon > 0$ small enough there is $\delta > 0$ such that if $r(x) < r(x^*) + \epsilon$, then $d(x, x^*) < \delta$. As $r(x^{(k)}) \to r^*$, we conclude that infinitely many elements of $x^{(k)}$ are in every neighborhood of x^*. As further $d(x^{(k+1)}, x^{(k)}) \to 0$, we have that $x^{(k)} \to x^*$, which contradicts the assumption that $x^{(k)}$ does not converge.

3. $r(x^{(k)}) \to r' = r^*$ and $x^{(k)}$ converges.

   This case must hold as all other cases are shown to be impossible, which completes the proof.

□

Notice, however, that Theorem 3.25 does not state the convergence of Algorithm 1 in general. Such a result remains an open question.

3.1.5.2 Scenario Sampling

As in Section 3.1.4, we now consider RecFeas in the center case with an infinite uncertainty set $\mathcal{U}$. Assuming that none of the presented reduction approaches apply, we can still find a heuristic solution to RecFeas by sampling a finite subset of scenarios. Algorithm 2 summarizes this approach.

Algorithm 2 (Sampling RecFeas)

```
Require: An uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$, a distance measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and a sample size $N \in \mathbb{N}$.
1: Draw a set $\mathcal{U}'$ of $N$ scenarios from $\mathcal{U}$.
2: Solve RecFeas($\mathcal{U}'$). Let $x$ be an optimal solution.
3: return $x$.
```

This is motivated by the sample average approximation method (see Section 2.8) from stochastic programming, where convergence is assured due to the law of large numbers. However, in robust optimization, this is usually not the case: Consider the function

$$f(x) = \begin{cases} 1 & x \in \mathbb{Q} \\ 0 & x \in \mathbb{R} \setminus \mathbb{Q} \end{cases}$$

on the interval $[0, 1]$. The expected value for f is 0 and will be found by sampling, but not the maximum value of 1. An example of the same structure can be constructed to show that sampling scenarios for RecFeas will not necessarily converge to the optimal solution.

As the median with respect to an infinite set of scenarios is not necessarily finite, we consider only the center case for RecFeas here. We analyze cases where sampling a sufficiently large number of scenarios does converge to an optimal solution of RecFeas. To do so, we make use of the following concepts from set-valued analysis, which can be found in any textbook on the topic, e.g. [AF90].

Definition 3.26. *Let $F : X \to 2^Y$ be a set-valued function, and let $dom(F) = \{x \in X : F(x) \neq \emptyset\}$.*

F is called lower hemicontinuous, if for every sequence $x_n \in X$ that converges to $x^* \in X$, and every $y^* \in f(x^*)$, there exists a sequence y_n such that $y_n \in F(x_n)$ for all $n \geq N$, and $y_n \to y^*$.

Let Δ be a metric on X. F is called upper hemicontinuous, if for all $x \in dom(X)$ and any neighborhood V of $F(x)$ there is $\epsilon > 0$ such that $F(x') \subseteq V$ for all x' with $\Delta(x, x') \leq \epsilon$.

A set-valued function is called continuous, if it is lower and upper hemicontinuous.

Definition 3.27. *A set-valued function $F : X \to 2^Y$ has a closed graph if $Gr(F) = \{(x, y) \in X \times Y : y \in F(x)\}$ is a closed subset of $X \times Y$.*

Note that a set-valued function F has a closed graph if and only if it satisfies the following condition: If x_n and y_n are sequences in X and Y such that $y_n \in F(x_n)$ for all n, $x_n \to x$ and $y_n \to y$, then $y \in F(x)$. We will further use the following theorems, which can also be found e.g. in [AF90].

Theorem 3.28. *Any upper-hemicontinuous set-valued function $F : X \to 2^Y$ has a closed graph. The converse is also true if Y is compact.*

If we consider the set of feasible solutions $\mathcal{F}$ with respect to ξ as a set-valued mapping from $\mathcal{U}$ to $\mathbb{R}^m$, we can now conclude:

Theorem 3.29. *Let an uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ be given, and let $\mathcal{F} : \xi \mapsto \mathcal{F}(\xi)$ be upper hemicontinuous. Let $\mathcal{U}'$ be a dense subset of $\mathcal{U} \subseteq \mathbb{R}^m$. Then $r(x,\mathcal{U}') = r(x,\mathcal{U})$ for all $x \in \mathcal{X}$.*

Proof. Let x be given, and let $\xi \in \mathcal{U}$ be such that $r(x,\mathcal{U}) = d(x, \mathcal{F}(\xi))$. Let Δ be a metric on $\mathbb{R}^m$. Then, for all $\epsilon > 0$ there is $\delta > 0$ such that

$$\mathcal{F}(\xi') \subseteq B_\epsilon(\mathcal{F}(\xi)) \; \forall \xi' \in B_\delta(\xi),$$

where $B_\epsilon(\mathcal{F}(\xi)) = \{x' \in \mathcal{X} : d(x', \mathcal{F}(\xi)) \leq \epsilon\}$, $B_\delta(\xi) = \{\xi' \in \mathcal{U} : \Delta(\xi', \xi) \leq \delta\}$. Therefore, $d(x, \mathcal{F}(\xi')) \geq d(x, \mathcal{F}(\xi)) - \epsilon$ for all $\xi' \in B_\delta(\xi)$. On the other hand, we have $d(x, \mathcal{F}(\xi)) \geq d(x, \mathcal{F}(\xi'))$ for all $\xi' \in \mathcal{U}$. As $\mathcal{U}'$ is dense in $\mathcal{U}$, for all δ there is a scenario $\xi' \in B_\delta(\xi) \cap \mathcal{U}'$. Hence, as $\epsilon \to 0$, we get $r(x,\mathcal{U}') = r(x,\mathcal{U})$. □

We now check if this theorem can be applied to the simple case of linear functions.

Lemma 3.30. *The set-valued function* $\mathcal{F} : \mathbb{R}^n \times \mathbb{R} \to 2^{\mathbb{R}^n}$, $(a, b) \mapsto \{x \in \mathbb{R}^n : a^t x \leq b\}$ *has a closed graph.*

Proof. Let (a_k, b_k) be a sequence with $(a_k, b_k) \to (a, b)$, and let $x_k \to x$ be such that $a_k^t x_k \leq b_k$. Set $c_n = b_n - a_n^t x_n \geq 0$ for all n. As $a_n^t x_n$ is the sum of the product of converging sequences, c_n is convergent. Consequently, $c_n \to c \geq 0$, and $a^t x \leq b$. □

Therefore, due to Theorem 3.28 we have that the function $\mathcal{F} : \mathbb{R}^n \times \mathbb{R} \to 2^{\mathbb{R}^n}$, $(a, b) \mapsto \{x \in \mathbb{R}^n : a^t x \leq b\} \cap \mathcal{X}$, where $\mathcal{X}$ is a compact set, is upper hemicontinuous. Also, the intersection of upper hemicontinuous set-valued functions is upper hemicontinuous again [AF90].

Note that Lemma 3.30 easily extends to continuous functions $a(\xi), b(\xi)$. We can hence state the convergence of the sampling approach for uncertain linear programs:

Corollary 3.31. *Let an uncertain optimization problem* $(P(\xi), \xi \in \mathcal{U})$ *with uncertain linear constraints be given, and let all constraints depend continuously on* ξ. *Let* $\mathcal{X}$ *be a compact set. Let* $(\mathcal{U}(N))$ *be a sequence of subsets of* $\mathcal{U} \subseteq \mathbb{R}^m$, *and* Δ *a metric on* $\mathbb{R}^m$. *If for all* $\xi \in \mathcal{U}$ *and* $\epsilon > 0$ *there is a number* K *with* $\Delta(\xi, \mathcal{U}(N)) \leq \epsilon$ *for all* $N \geq K$, *then* $r(x, \mathcal{U}(N)) \to r(x, \mathcal{U})$ *for all* $x \in \mathcal{X}$.

Corollary 3.32. *Let the requirements of Corollary 3.31 be fulfilled. Then for all* $x \in \mathcal{X}$

$$r(x, \mathcal{U} \setminus \{\xi^1, \ldots, \xi^S\}) = r(x, \mathcal{U})$$

for any set $\{\xi^1, \ldots, \xi^S\} \subseteq \mathcal{U}$ *of scenarios with* $\mathcal{U} \setminus \{\xi^1, \ldots, \xi^S\}$ *is dense in* $\mathcal{U}$.

In fact, Corollary 3.31 is not restricted to uncertain problems with linear constraints and continuous uncertainty, but can be extended to any continuous constraints F.

Lemma 3.33. *Let* $F : \mathbb{R}^n \times \mathbb{R}^n \to \mathcal{X}$, *where* $\mathcal{X} \subseteq \mathbb{R}$ *is a compact set, be continuous. Then the set-valued function* $F' : \mathbb{R}^n \to 2^{\mathcal{X}}$, $F'(y) = \{x \in \mathbb{R}^n : F(x, y) \leq 0\}$ *is upper-hemicontinuous.*

Proof. We show that F' has a closed graph. Let $x_n \to x$ and $y_n \to x$ be convergent sequences with $x_n \in F'(y_n)$ for all n. Then, $F(x_n, y_n) \leq 0$ and (x_n, y_n) is a convergent sequence. Due to the continuity of F, we therefore have $F(x, y) \leq 0$, i.e., $x \in F'(y)$. $\square$

Note that if a function $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R},\ (x, y) \mapsto f(x, y)$ is continuous both the x and the y component, it is also (jointly) continuous.

3.1.6 Summary and Conclusion

Table 3.1 summarizes the non-algorithmic results obtained in this section.

uncertainty set $\mathcal{U}$	constraints $F(\cdot, \xi)$	uncertainty $F(x, \cdot)$	rec. costs d	implementable decisions $\mathcal{X}$	results
finite	quasiconvex	arbitrary	norm	convex and closed	- RecFeas($\mathcal{U}$) convex problem (Lemma 3.7)
				$\mathcal{X} = \mathbb{R}^n$	- Reduction to RecFeas($\bar{\mathcal{U}}$) for smaller sets $\bar{\mathcal{U}}$ (Theorem 3.8)
finite	linear	arbitrary	block norm	polyhedron	- RecFeas($\mathcal{U}$) linear problem (Theorem 3.10)
polyhedron	bounded gradients	quasiconvex	norm	closed	- solution w.r.t extreme points of $\mathcal{U}$ is reliable (Theorem 3.14)
polyhedron	jointly quasiconvex		norm	closed	- solution w.r.t. extreme points of $\mathcal{U}$ is optimal (Theorem 3.15)
polyhedron	convex	quasiconvex, right-hand side	norm	closed	- solution w.r.t extreme points of $\mathcal{U}$ is optimal (Corollary 3.16)
				convex and closed	- RecFeas($\mathcal{U}$) convex problem (Corollary 3.17)
polyhedron	linear	quasiconvex, right-hand side	block norm	polyhedron	- RecFeas($\mathcal{U}$) linear problem (Theorem 3.18)

Table 3.1 Summary of properties for the RecFeas($\mathcal{U}$) depending on the optimization problem P(ξ), the uncertainty set $\mathcal{U}$, the type of uncertainty, and the recovery costs.

Concerning the algorithmic results on RecFeas, we were able to show convergence of the iterative procedure, and convergence of the sampling approach under certain circumstances.

3.2 RecOpt

3.2.1 Introduction

In the previous section, we analyzed the concept of recovering a robust solution to be feasible in any scenario, minimizing the recovery distance given by a function d, i.e.,

$$\min_{x\in\mathcal{X}}\sup_{\xi\in\mathcal{U}} d(x,\mathcal{F}(\xi))$$

So far we have ignored the nominal objective, which we may consider as an additional side constraint, when we fix a minimal objective quality. Recovering not just to any feasible solution, but even to an optimal one, is the focus of this section.

It is structured as follows: In the following section, we formally introduce the model of recovery-to-optimality RecOpt. We consider a simplified sampling approach in Section 3.2.3 and consider cases where an infinite uncertainty set can be reduced to a finite set of scenarios in Section 3.2.4. Finally, we discuss the issue of nominal feasibility of a RecOpt solution in Section 3.2.5, and conclude the analysis of RecOpt in Section 3.2.6.

3.2.2 The Model: Recovery-to-Optimality

Let an uncertain optimization problem $(P(\xi),\xi\in\mathcal{U})$ be given. For every $\xi\in\mathcal{U}$, denote by $f^*(\xi)$ the optimal objective value in the respective scenario. The recovery-to-optimality counterpart (RecOpt) of the uncertain problem is then given by:

$$\begin{aligned}
\text{(RecOpt)}\qquad & \min_{x\in\mathcal{X}}\max_{\xi\in\mathcal{U}} d(x,x^\xi)\\
\text{s.t.}\quad & x^\xi\in\mathcal{F}(\xi)\ \forall\xi\in\mathcal{U}\\
& f(x^\xi,\xi)=f^*(\xi)\ \forall\xi\in\mathcal{U}\\
& x\in\mathcal{X}, x^\xi\in\mathcal{X}\ \forall\xi\in\mathcal{U}
\end{aligned}$$

As before, we may also formulate RecOpt as a median problem instead of a center problem. Figure 3.7 sketches the idea of the concept. We are given a finite uncertainty set $\mathcal{U}=\{\xi^1,\xi^2,\xi^3\}$, and $x(\xi^i)$ denotes the unique optimal solution for scenario ξ^i, $i=1,2,3$. Similar to RecFeas, this approach does not require the set of strictly robust solutions $\mathbb{SR}$ to be non-empty, but it will always output some solution (if the single scenarios are feasible). Note that even if $\mathbb{SR}\neq\emptyset$, the outcome of RecOpt need not be contained in $\mathbb{SR}$ as it is shown in Figure 3.7, which is different for RecFeas. This is an advantage in many applications, since the objective value of the solution which will be realized after the scenario becomes known is in these cases much better than the objective value of a strictly robust solution.

In both RecFeas and RecOpt, we minimize the distance to sets. In fact, if we relax the constraint

$$f(x^\xi,\xi)=f^*(\xi)\ \forall\xi\in\mathcal{U}$$

to

$$f(x^\xi,\xi)\le(1+\rho)f^*(\xi)\ \forall\xi\in\mathcal{U}$$

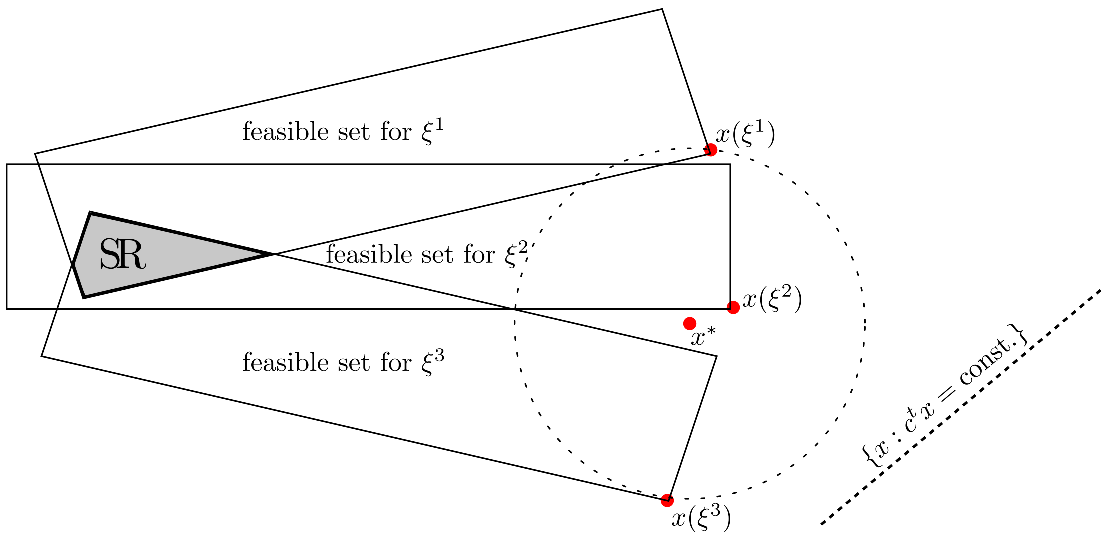


Figure 3.7 The output x^* of RecOpt using the Euclidean center is far away from the set of strictly robust solutions SR; it hence has a much better objective value.

(RecOpt) and (RecFeas) become equivalent for a budget ρ that is large enough. Therefore, (RecOpt) can be seen as (RecFeas) with additional constraints on the objective value quality of the feasible solutions we recover to, i.e., we consider the recovery distance to the feasible sets

$$\mathcal{F}'(\xi) = \{x \in \mathcal{F}(\xi) : f(x,\xi) \leq (1+\rho)f^*(\xi)\}$$

Note that in particular, the results on both the iterative Algorithm 1 and the sampling Algorithm 2 hold in this case. Note that $\{f(x,\xi) \leq (1+\rho)f^*(\xi)\}$ is a convex set for convex objective functions f.

Note that in order to apply Corollary 3.31, we need to show that $f^*(\xi)$ is a continuous function in ξ, if it is bounded in every scenario. To see this, we can proceed as follows: We can show that linear constraints $ax \leq b$ are lower hemicontinuous in (a,b); and as the intersection of lower hemicontinuous maps is lower hemicontinuous again, if the intersection is non-empty and both maps have non-empty interior [AF90], we can conclude that the uncertain feasible polyhedron $\mathcal{F}(\xi)$ is continuous in ξ. Applying the Maximum Theorem for set-valued functions then yields the continuity of f.

However, there is also a much simpler way if we make use of duality:

Assume that $LP(\xi)$ is given as

$$\begin{aligned} \max\ & c(\xi)^t x \\ & A(\xi)x \leq b(\xi) \\ & x \geq 0 \end{aligned}$$

If we assume that the optimal objective value is bounded in every scenario, we can use strong duality and consider the following problem $LP'(\xi)$:

$$\begin{aligned}\max\ & 0\\ & c(\xi)^t x = b^t(\xi) y\\ & A(\xi)x \le b(\xi)\\ & A^t(\xi)y \ge c(\xi)\\ & x, y \ge 0\end{aligned}$$

Because the feasible set is given by upper hemicontinuous functions, we conclude that the sampling approach converges, if w.l.o.g we intersect the feasible sets of LP' with a compact set $\mathcal{X} \subset \mathbb{R}^{n+m}$.

We consider a simple example. Let an uncertain linear optimization problem $(LP(\xi), \xi \in \mathcal{U})$ be given, where $\mathcal{U} = [0, 1]$ and

$$\begin{aligned} LP(\xi) \quad & \max\ \xi x_1 + (1-\xi)x_2\\ & \text{s.t.}\ \ x_1 + x_2 \le \frac{3}{2}\\ & \qquad 0 \le x_1, x_2 \le 1\end{aligned}$$

For the set of optimal solutions $opt(\xi)$, we get

$$opt(\xi) = \begin{cases} \operatorname{conv}\{(0,1),(0.5,1)\} & \text{if } \xi = 0\\ \{(0.5,1)\} & \text{if } \xi \in]0, 0.5[\\ \operatorname{conv}\{(0.5,1),(1,0.5)\} & \text{if } \xi = 0.5\\ \{(1,0.5)\} & \text{if } \xi \in]0.5, 1[\\ \operatorname{conv}\{(0.5,1),(1,1)\} & \text{if } \xi = 1 \end{cases}$$

The cases $0 < \xi < 0.5$, $\xi = 0.5$ and $0.5 < \xi < 1$ are depicted in Figure 3.8. Note that sampling $\mathcal{U}$ uniformly has a probability of 0 to find the case $\xi = 0.5$. In particular, the sampling approach does not "see" the optimal solutions $\operatorname{conv}\{(0.5, 1), (1, 0.5)\}$.

However, *opt* is a lower hemicontinuous set-valued function, and $d(x, \mathcal{F}(0.5)) \le d(x, \mathcal{F}(\xi))$ for all ξ in a neighborhood of 0.5. Therefore,

$$r(x,]0, 0.5[\cup]0.5, 1[) = r(x, [0, 1]).$$

3.2.3 A Simplified Sampling Approach to RecOpt

We now consider a solution approach to RecOpt that determines just a single optimal solution to every scenario instead of the whole set. Note that this approach is optimal, if the optimal solution in every scenario is unique.

It has the advantage that it can be applied whenever a solution algorithm $\mathfrak{A}$ for the original optimization problem is at hand, and significantly reduces the problem complexity. The procedure is the following: First, choose a finite set of scenarios from

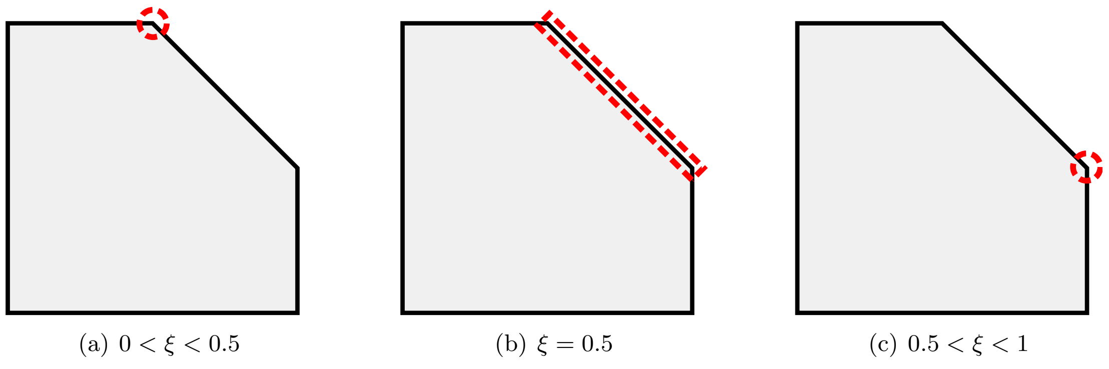


Figure 3.8 Optimal sets, depending on objective function $\xi x_1 + (1-\xi)x_2$.

the uncertainty set $\mathcal{U}$. Then use the algorithm $\mathfrak{A}$ to generate an optimal solution for each of these scenarios. These solutions can be regarded as a set of points $\mathcal{S} \subseteq \mathbb{R}^n$. In the third step we calculate one point $x \in \mathcal{X}$ which represents the set $\mathcal{S}$ by solving a location problem.

Therefore, RecOpt reduces to a (classic) location problem where the given set of facilities are singletons. For a finite set of points $\{x^1, \ldots, x^N\}$, we denote this location problem as

$$\text{(Loc)} \qquad \min_{x \in \mathcal{X}} \quad \text{loc}(x) = h(d(x, x^1), \ldots, d(x, x^N))$$

where $\mathcal{X}$ is the decision space and h is a function that is usually monotonically increasing in each of its arguments. In particular, for the median problem we have $\text{loc}(x) = \sum_{i=1}^N d(x^i, x)$, and for the center problem $\text{loc}(x) = \max_{i=1}^N d(x^i, x)$. Note that minimizing $\text{loc}(x) = \sum_{i=1}^N d^2(x^i, x)$ for the Euclidean distance $d = l_2$ can be done easily by taking the average of every coordinate and leads to the center of gravity of $x^1, \ldots, x^N$, the *centroid.*

RecOpt can now be heuristically solved using Algorithm 3.

Algorithm 3 (Singleton Sampling RecOpt)

```
Require: An uncertain problem $P(\xi)$ with uncertainty set $\mathcal{U}$, and an algorithm $\mathfrak{A}$ for
    solving $P(\xi)$ if the data $\xi \in \mathcal{U}$ is known.
 1: Choose a set of scenarios $\{\xi^1, \ldots, \xi^N\} \subseteq \mathcal{U}$.
 2: Use $\mathfrak{A}$ to solve $P(\xi^i)$ for all $i = 1, \ldots, N$. Let $x^i$, $i = 1, \ldots, N$ be the resulting
    optimal solutions.
 3: Find a robust solution $x^*$ by solving a location problem (Loc) in which the existing
    points are $x^1, \ldots, x^N$.
 4: return $x^*$.
```

In the following we investigate RecOpt for norm-based distance functions d given as $d(x, y) = \|y - x\|$ for some norm $\|\cdot\|$.

We now discuss some properties of the robust solution obtained by Algorithm 3. Suppose that the optimal solutions $x^1, \dots, x^N$ all satisfy some property, or lie within some specific set. This may be the case, e.g., if some constraints of the problem $\mathrm{P}(\xi)$ are certain, or if they are likely to hold. We show that, under some conditions, the output x^* of Algorithm 3 also satisfies this property. We denote a distance d to be *linear equivalent* to the Euclidean distance l_2 if for all $x, y \in \mathbb{R}^n$ it holds $d(x,y) = l_2(Tx, Ty)$ for a regular linear transformation T.

Theorem 3.34. *Let $x^1, \dots, x^N$ be optimal solutions to the scenarios $\xi^1, \dots, \xi^N$ chosen in Algorithm 3 and let x^* be the output of Algorithm 3. Let $Q(x^i) \le \delta$, $i = 1, \dots, N$, for some convex function $Q : \mathbb{R}^n \to \mathbb{R}$, $\delta \in \mathbb{R}$. Then*

$$Q(x^*) \le \delta$$

if the objective function $\mathrm{loc}(x)$ that has to be minimized in (Loc) in step 3 takes the following form:

1. $\mathrm{loc}(x) = \sum_{i=1}^N d^2(x^i, x)$, where d^2 is the squared Euclidean distance.
2. $\mathrm{loc}(x) = \sum_{i=1}^N d(x^i, x)$, where d is linear equivalent to the Euclidean distance.
3. $\mathrm{loc}(x) = \max_{i=1}^N d(x^i, x)$, where d is linear equivalent to the Euclidean distance.
4. $\mathrm{loc}(x) = \sum_{i=1}^N d(x^i, x)$, where d is any l_p-norm for $1 < p < \infty$ and $n = 2$.

Proof. In case 1, x^* is the center of gravity, i.e., $x^* = \frac{1}{N}\sum_{i=1}^N x^i$. Hence x^* lies in the convex hull of the existing facilities. For the Euclidean norm and any norm which is linear equivalent to the Euclidean norm, it is shown in [Pla84] that the same property holds in the cases stated in 2. and 3., i.e., $x^* \in \mathrm{conv}(\{x^1, \dots, x^N\})$. For case 4, the same holds due to [JL83]. We thus find positive λ_i, $i = 1, \dots, N$ with $\sum_{i=1}^N \lambda_i = 1$ and $x^* = \sum_{i=1}^N \lambda_i x^i$. From this we obtain

$$Q(x^*) = Q\left(\sum_{i=1}^N \lambda_i x^i\right) \le \sum_{i=1}^N \underbrace{\lambda_i}_{\ge 0} \underbrace{Q(x^i)}_{\le \delta} \le \delta \sum_{i=1}^N \lambda_i = \delta.$$

□

The proof is based on the property that $x^* \in \mathrm{conv}\{x^1, \dots, x^N\}$, which does in general only hold for the cases mentioned in the theorem. For $n > 2$ and d not linear equivalent to the Euclidean distance, counterexamples to this property can be constructed (see [Pla84]) and hence Theorem 3.34 does not hold in these cases. The following consequences follow directly from the theorem and are useful properties of the solutions obtained by Algorithm 3.

Corollary 3.35. *Let an uncertain linear optimization problem $(LP(A,b,c),(A,b,c) = \xi \in \mathcal{U})$ be given. Let $x^1,\dots,x^N$ be optimal solutions to the scenarios $\xi^1,\dots,\xi^N$ chosen in Algorithm 3 and let x^* be the output of Algorithm 3. Let $\operatorname{loc}(x)$ be as described in Theorem 3.34. Then we have:*

1. *If the objective function value of all x^i is better than α then also the objective value of x^* is better than α.*
2. *If some of the constraints of LP(A,b,c) are certain, i.e., if $d^t x^i \leq \delta_i$ (or $d^t x^i = \delta_i$) holds for $i = 1,\dots,N$ then also $d^t x^* \leq \delta$ (or $d^t x^* = \delta$).*
3. *If F is a convex set and $x^i \in F$ for all $i = 1,\dots,N$ then it is sufficient to solve*
$$\min_{x\in\mathbb{R}^n} h(d(x^1,x),\dots,d(x^N,x))$$
instead of $\min_{x\in F} h(d(x^1,x),\dots,d(x^N,x))$ in Step 3.

Lemma 3.36. *Let $(P(b))$ be an uncertain problem with constraints $Ax \geq b$ only depending on the right-hand side. Let $\hat{b} \in \mathcal{U}$ be the nominal scenario with $\hat{b} \leq b$ for all $b \in \mathcal{U}$. Let $S \subseteq \mathcal{U}$ be a finite set and let x^b be an optimal solution to $(P(b))$ for all $b \in S$. Then the centroid, i.e. the solution to $\min_x \sum_{b\in S} \|x - x^b\|_2^2$, is feasible for the nominal scenario.*

Proof. Let $x \in \mathbb{R}^n$ be the centroid. For the kth constraint, we obtain:

$$\sum_{i=1}^n a_{ki}x_i = \sum_{i=1}^n a_{ki}\frac{1}{|S|}\sum_{b\in S} x_i^b = \frac{1}{|S|}\sum_{b\in S}\sum_{i=1}^n a_{ki}x_i^b \geq \frac{1}{|S|}\sum_{b\in S} b_k \geq \frac{1}{|S|}\sum_{j=1}^{|S|}\hat{b}_k = \hat{b}_k$$

□

This result naturally extends to interval-based uncertainties of the form $[\hat{b}-\epsilon, \hat{b}+\delta]$ with $\delta > \epsilon$, i.e., the nominal scenario does not need to be the smallest one.

3.2.4 Reduction Approaches for RecOpt

We now discuss how to choose a finite set of scenarios for Step 1 of Algorithm 3. If the uncertainty set $\mathcal{U}$ is finite (and not too large) we might be able to solve $\mathrm{P}(\xi)$ for all $\xi \in \mathcal{U}$. But how to proceed if $\mathcal{U}$ contains a very large or even infinite number of possible scenarios?

The easiest answer is to run RecOpt with a large set of randomly generated scenarios $\xi^1,\dots,\xi^N$ in this case and proceed with Steps 2 and 3 as before. The outcome is an *approximate* solution whose quality is numerically investigated in Section 5.4.6. Note that Algorithm 2 converges to the optimal solution of RecFeas for an increasing sample size. Therefore, if the solution algorithm $\mathfrak{A}$ returns a dense subset of optimal solutions when repeatedly applied to the same scenario, we can conclude that Algorithm 3 also converges to an optimal solution of RecOpt. We now consider cases where we do not need to sample a dense subset of scenarios to solve RecOpt to optimality.

In the following we consider the center problem

$$(\text{Loc}) \qquad \min_{x \in \mathcal{X}} \quad \text{loc}(x) = \max \left\{ d(x^1, x), \dots, d(x^N, x) \right\}.$$

Theorem 3.37. *Let $\mathcal{U} = \text{conv}\{\xi^1, \dots, \xi^N\} \subseteq \mathbb{R}^M$ and let $d(x, \cdot)$ be convex in its second argument. Let $x : \mathbb{R}^M \to \mathbb{R}^n$ assign an optimal solution $x(\xi)$ to any scenario ξ, and assume that x is affine linear. By writing $x^i := x(\xi^i)$ for short we have*

1. *For all $\xi \in \mathcal{U}$: $x(\xi) \in \text{conv}\{x^1, \dots, x^N\}$.*
2. *The center of $x^1, \dots, x^N$ with respect to the distance measure d solves RecOpt.*

Proof. Let $\xi \in \mathcal{U}$, i.e. there exist λ_i, $i = 1, \dots, N$ with $0 \leq \lambda_i \leq 1$, $\sum_{i=1}^N \lambda_i = 1$ and $\xi = \sum_{i=1}^N \lambda_i \xi^i$. Then we obtain

$$x(\xi) = x\left(\sum_{i=1}^N \lambda_i \xi^i\right) = \sum_{i=1}^N \lambda_i x(\xi^i) = \sum_{i=1}^N \lambda_i x^i,$$

i.e. $x(\xi) \in \text{conv}\{x^1, \dots, x^N\}$. Concerning the second part of the theorem, define $r^* := \max_{i=1,\dots,N} d(x^*, x^i)$ as the radius of the center x^* and let $\bar{r}$ be the best possible objective value for RecOpt. Since $r^* \leq \bar{r}$ it remains to show that the recovery radius of x^* with respect to $\mathcal{U}$ equals r^*, i.e. that $d(x^*, x(\xi)) \leq r^*$ for all $\xi \in \mathcal{U}$.

To this end, let $\xi \in \mathcal{U}$. Then $x(\xi) \in \text{conv}\{x^1, \dots, x^N\}$ and hence there are λ_i, $i = 1, \dots, N$, with $0 \leq \lambda_i \leq 1$, $\sum_{i=1}^N \lambda_i = 1$ and $\sum_{i=1}^N \lambda_i x^i = x(\xi)$. Quasi-convexity of $d(x^*, \cdot)$ yields

$$d(x^*, x(\xi)) = d(x^*, \sum_{i=1}^N \lambda_i x^i) \leq \max_{i=1}^N d(x^*, x^i) = r^*,$$

hence x^* is in fact optimal for RecOpt. □

This raises the question, when the solution mapping x is indeed affine linear. We present some results on general linear programs with uncertain right-hand side, i.e.

$$(P(b)) \quad \min\{c^t x : Ax = b, x \geq 0, x \in \mathbb{R}^n\}, b \in \mathcal{U}, \tag{3.9}$$

where $A \in \mathbb{R}^{m \times n}$.

Lemma 3.38. *Consider (P(b)) with a convex uncertainty set $\mathcal{U} \subseteq \mathbb{R}^M$ and assume that int($\mathcal{U}$) $\neq \emptyset$, where int($\mathcal{U}$) denotes the interior of $\mathcal{U}$. Then $x : \mathbb{R}^M \to \mathbb{R}^n$ as defined in Theorem 3.37 is an affine linear function if and only if there exists a basis $B \subseteq \{1, \dots, n\}$ with non-negative reduced costs and $A_B^{-1} b \geq 0$ for all $b \in \mathcal{U}$.*

Proof.

- "if": Let B be such a basis. Since the reduced costs $c_n^t - c_B^t A_B^{-1} A_n \geq 0$ are independent of b and feasibility of the corresponding basic solution is ensured for all $b \in \mathcal{U}$ we know from linear programming theory that $x(b) := (A_B^{-1} b, 0)$ is optimal for (P(b)). Hence, $x(b)$ is an affine linear function.

- "only if": Choose any $b^0 \in \text{int}(\mathcal{U})$ and solve the linear program. This yields a basis B with non-negative reduced costs and $A_B^{-1} b^0 \geq 0$, i.e. $x(b^0) = (A_B^{-1} b^0, 0)$ is an optimal solution.

  As $b^0 \in \text{int}(\mathcal{U})$ we can find for every unit vector $e_i \in \mathbb{R}^M$ an ϵ_i and a direction $d_i \in \{-1, +1\}$ such that
  $$b^i := b^0 + \epsilon_i d_i e_i \in \mathcal{U}$$
  and $A_B^{-1} b^i \geq 0$. Hence, B is an optimal basis for $b^0, b^1, \ldots, b^M$, i.e. we have $x(b^i) = (A_B^{-1} b^i, 0)$ for $i = 0, 1, \ldots, M$. Due to our assumption $x(b)$ is affine linear; hence it is uniquely determined on the set of $\{b^0, b^1, \ldots, b^M\}$ of $M+1$ affinely independent points. This yields $x(b) = (A_B^{-1} b, 0)$ for all $b \in \mathcal{U}$, in particular we have $A_B^{-1} b \geq 0$ for all $b \in \mathcal{U}$.

□

We now investigate (P(b)) with the additional assumption that the uncertainty set $\mathcal{U} \subseteq \mathbb{R}^m$ (in this case, $M = m$) is symmetric with respect to some specified vector $b^* \in \mathbb{R}^m$, that is, for all $b \in \mathcal{U}$ there is a $\hat{b} \in \mathcal{U}$, such that $b - b^* = b^* - \hat{b}$. We show that in this case b^* solves RecOpt. To this end, we first need the following lemma about the center of a symmetric location problem.

Lemma 3.39. *Let $C \subseteq \mathbb{R}^n$ be a compact set of points that is symmetric with respect to $x^* \in \mathbb{R}^n$. Let d be a distance measure that has been derived from a norm, i.e. $d(x, y) = \|y - x\|$ for some norm $\|\cdot\|$. Then x^* is a center of C.*

Proof. Let $\max_{x \in C} d(x, x^*) = r$ and let $y_1, y_2 \in C$ be a pair of symmetric points (i.e. $y_1 - x^* = x^* - y_2$) that maximizes the distance to x^*. Let x' be any point. Applying the triangle inequality and using that y_1, x^*, y_2 are collinear yields
$$2r = d(y_1, x^*) + d(x^*, y_2) = d(y_1, y_2) \leq d(y_1, x') + d(x', y_2)$$
and therefore either $r \leq d(y_1, x')$ or $r \leq d(x', y_2)$ holds. We conclude that
$$\max_{x \in C} d(x, x') \geq \max\{d(y_1, x'), d(y_2, x')\} \geq r,$$
hence x' cannot be better than x^*. □

Theorem 3.40. *Let (P(b)), $b \in \mathcal{U}$ be an uncertain linear program (3.9) and let $\mathcal{U}$ be symmetric with respect to $b^* \in \mathbb{R}^m$. Let B be an optimal basis for (P(b^*)) and assume that $A_B^{-1} b \geq 0$ for all $b \in \mathcal{U}$. Then $x(b^*)$ solves RecOpt.*

Proof. B is an optimal basis for every $b \in \mathcal{U}$, as $A_B^{-1}b \geq 0$. Thus $x(b) = A_B^{-1}b$. As $\mathcal{U}$ is a symmetric set with respect to b^* and x an affine linear mapping, the set of optimal solutions is symmetric with respect to $x(b^*)$ and we can apply Lemma 3.39. □

This directly gives a result for all interval-based uncertainty sets.

Corollary 3.41. *Let (P(b)), $b \in \mathcal{U} = \{b \in \mathbb{R}^m : \underline{\eta} \leq b \leq \overline{\eta}\}$ be an uncertain linear program (3.9) and let $\underline{\eta}, \overline{\eta} \in \mathbb{R}^m$. Let $b \in \mathcal{U}$ and let B be an optimal basis for (P(b)). If $A_B^{-1}\underline{\eta} \geq 0$ and $A_B^{-1}\overline{\eta} \geq 0$ both hold, then an optimal solution of RecOpt can be found by solving $(P(b^*))$ with $b^* := \frac{\underline{\eta}+\overline{\eta}}{2}$.*

3.2.5 Nominal Feasibility

So far, we have ignored feasibility issues when determining the solution to RecOpt. For mixed-integer linear programs, feasibility in the nominal case can be simply added as a set of constraints to (Loc). However, the resulting location problem may become computationally challenging, and algorithms for the nominal problem are usually not applicable. Especially if there is a good black-box algorithm available for the nominal problem, it would be preferable to avoid this situation. In this section, we present a class of problems in which this is possible.

Consider an optimization problem with a big-M constraint of the form

$$\sum_{i=1,\dots,n} a_i x_i \leq My$$

where x is continuous, and y is binary. Even though solving a location problem with respect to a set of feasible solutions for this constraint may result in a solution (x^*, y^*) with a fractional value for y^*, it is always possible to complete x^* to a feasible solution. We formalize this concept:

Definition 3.42. *Let P be an optimization problem, and let $(\mathcal{X}_1, \mathcal{X}_2)$ be a partition of the problem variables. We call P* combinable *with respect to the variables $\mathcal{X}_1$, if*

$$x_1 \in Pr_{\mathcal{X}_1}(\mathcal{F})$$

for all $x_1 \in \operatorname{conv}\{x_1^1, \dots, x_1^N\}$ and all sets of feasible solutions $\{(x_1^1, x_2^1), \dots, (x_1^N, x_2^N)\}$ to P.

Note that the idea of Definition 3.42 is that the variables $\mathcal{X}_2$ can determined in a feasible way, once the variables $\mathcal{X}_1$ are fixed.

Theorem 3.43. *Let $(P(\xi), \xi \in \mathcal{U})$ be an uncertain optimization problem with finite uncertainty, and let $P(\hat{\xi})$ be combinable with respect to the variables x_1. Let $\mathcal{F}(\xi) \subseteq \mathcal{F}(\hat{\xi})$ for all $\xi \in \mathcal{U}$. Let d be a distance measure of any of the types from Theorem 3.34 that only depends on the variables x_1. Then there is a solution to RecOpt that is feasible for the nominal scenario.*

Proof. Let $\{x^1, \ldots, x^N\}$ be optimal solutions to the scenarios $\{\xi^1, \ldots, \xi^N\}$, respectively. Let (x_1^*, x_2^*) be an optimal solution to the location problem (Loc). Due to Theorem 3.34, we have $x_1^* \in \operatorname{conv}\{x_1^1, \ldots, x_1^N\}$. As P is combinable, there is an x_2' such that (x_1^*, x_2') is feasible for the nominal problem. As d does not depend on the variables $\mathcal{X}_2$, (x_1^*, x_2') has the same objective value for (Loc) as (x_1^*, x_2^*), which completes the proof. $\square$

Theorem 3.34 especially applies to mixed-integer problems in which the discrete variables are used for modeling constraints, and can be neglected for the recovery distance measure. Even though we ignore them in the location problem, we can complete its solution with feasible discrete values again.

Examples for combinable problems:

All constraints of the form $\sum a_i x_i \leq b$ and $\sum a_i x_i \geq b$ are combinable with respect to all variable subsets. The only problematic cases are equality constraints. As an example, consider a constraint of the form $x_1 + 10x_2 = 5$, where $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{Z}$, and the feasible solutions $(5, 0)$ and $(-5, 1)$. Setting $x_1 = 0$ cannot be completed to a feasible solution. The constraint $x_1 + x_2 + 10x_3 = 5$ with $x_1, x_2 \in \mathbb{R}$ and $x_3 \in \mathbb{Z}$, on the other hand, is combinable with respect to x_1. An example for a mixed-integer problem of this type is periodic timetabling - further details can be found in Section 5.3.

3.2.6 Conclusion

In this section we considered an extension of RecFeas, where an additional constraint on the nominal quality of a robust solution is added. Although the results on RecFeas still apply, we can now consider a simplified problem formulation, where we do not minimize the distance to a set of optimal solution, but just to a single solution we determined using any algorithm that can be applied to the nominal problem. Being computationally simpler, this approach is still optimal when the optimal solution of every scenario is unique. We derived properties of solutions generated this way, and determined further cases when such a solution is optimal, or nominally feasible.

3.3 Relations between Concepts

3.3.1 RecFeas and RecOpt in the Landscape of Robust Optimization

In this section we discuss the relationship between RecFeas and RecOpt to the current concepts of robust optimization. We begin with introducing two criteria to classify a robustness concept:

One-stage/Two-stage. A general difference between all presented robustness concepts is that some of them allow parts of the solutions to be updated when the parameter realization becomes known, while others do not allow this possibility.

Robustness measure. Some of the robustness concepts do not consider robustness as a "yes/no" criterion, but instead develop measures for the degree of robustness of a solution.

Applying this classification scheme to the concepts of Chapter 2, we get the following classification matrix:

	One-stage	Two-stage
robustness measure	light robustness, reliability, soft robustness, UFO	recovery robustness, RecFeas, RecOpt
no robustness measure	strict robustness, Bertsimas and Sim, regret optimization	adjustable robustness

Table 3.2 Robustness concepts.

Within this scheme, RecOpt and RecFeas are most similar to the concept of recovery robustness. We now discuss similarities and differences to the various concepts in detail.

Strict robustness. As shown in Section 3.1.2, RecFeas has an objective value of 0 if and only if there is a strictly robust solution, i.e.,

$$r(x,\mathcal{U}) = 0 \iff x \in \mathbb{SR}.$$

This relationship can be extended: Let any metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be given. We optimize lexicographically, by optimizing $d(x, y)$ first, and then $\max_{\xi\in\mathcal{U}} f(x, \xi)$. Then this approach is a direct extension of strict robustness in the sense that if the strictly robust counterpart is feasible, both solutions coincide, but it still yields a solution in the case of $\mathbb{SR} = \emptyset$.

Bertsimas and Sim. Consider again the case of an uncertain constraint of the form

$$a_1x_1 + \ldots + a_nx_n \leq b$$

with an uncertainty $\mathcal{U} = \{a \in \mathbb{R}^n : a_i \in [\hat{a}_i - d_i, \hat{a}_i + d_i], i = 1, \ldots, n\}$. Note that the robust counterpart in the sense of Bertsimas and Sim is equivalent to the strictly robust counterpart with respect to the uncertainty set

$$\mathcal{U}' = \{a \in \mathcal{U} : |\{i \in \{1, \ldots, n\} : a_i \neq \hat{a}_i\}| \leq \Gamma\}$$

Therefore, we have that a vector x is feasible for the Bertsimas and Sim robust counterpart if and only if $r(x, \mathcal{U}') = 0$.

Adjustable robustness. We can consider RecFeas as an adjustable robust counterpart. For a given uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ and distance measure $d(x, y)$, consider the uncertain problem

$$\begin{array}{lrl} P'(\xi) \qquad & \min & d(x, y) \\ & \text{s.t.} & x \in \mathcal{F}(\hat{\xi}) \\ & & y \in \mathcal{F}(\xi) \\ & & x, y \in \mathcal{X} \end{array}$$

Considering y as a vector of adjustable variables, the adjustable robust counterpart of $(P'(\xi), \xi \in \mathcal{U})$ is then given as

$$\begin{array}{lrl} \text{(adR)} \qquad & \min \max\limits_{\xi \in \mathcal{U}} & d(x, y^\xi) \\ & \text{s.t.} & x \in \mathcal{F}(\hat{\xi}) \\ & & y^\xi \in \mathcal{F}(\xi) \ \forall \xi \in \mathcal{U} \\ & & x, y^\xi \in \mathcal{X} \ \forall \xi \in \mathcal{U} \end{array}$$

which can be interpreted as the RecFeas counterpart of the original uncertain problem $(P(\xi), \xi \in \mathcal{U})$.

Recovery robustness. As already noted in the classification scheme, RecOpt and RecFeas have the most conceptual similarities to recovery robustness. The main difference is that the recovery approaches in this work contain the recovery algorithm only implicitly via the distance function d, and not explicitly via a set of possible recovery algorithms $\mathcal{A}$. What seems to be only a slight variation, offers the possibility to use methods and concepts from location theory, that result in a rigorous analytical discussion of solutions. In that sense the concepts derived in this work can be considered as an evolution of recovery robustness.

Regret optimization. We show that min max regret optimization can be interpreted as a recovery to optimality concept with a special distance function. We define

$$\Xi(y) = \{\xi \in \mathcal{U} : f(y, \xi) = \min_{y' \in \mathcal{X}} f(y', \xi)\}$$

and set

$$d(x,y) = \max_{\xi \in \Xi(y)} \left[f(x,\xi) - f(y,\xi)\right].$$

For our purposes, we may set any value for $d(x,y)$, if $\Xi(y) = \emptyset$. Then both concepts are equivalent:

Lemma 3.44. *For a given $x \in \mathcal{X}$, it holds*

$$\max_{\xi \in \mathcal{U}} \left[f(x,\xi) - f(x(\xi),\xi)\right] = \max_{\xi \in \mathcal{U}} d(x, x(\xi)).$$

Proof.

$$\begin{aligned} &\max_{\xi \in \mathcal{U}} d(x, x(\xi)) \\ =&\max_{\xi \in \mathcal{U}} \max_{\xi' \in \Xi(x(\xi))} \left[f(x,\xi') - f(x(\xi),\xi')\right] \\ =&\max_{\xi \in \mathcal{U}} \max_{\xi' \in \Xi(x(\xi))} \left[f(x,\xi') - f(x(\xi),\xi)\right] \end{aligned}$$

Assume that for a given $\xi \in \mathcal{U}$ there is a scenario $\xi' \in \Xi(x(\xi))$ such that

$$f(x,\xi') - f(x(\xi),\xi) \geq f(x,\xi) - f(x(\xi),\xi)$$

Then also

$$f(x,\xi') - f(x(\xi'),\xi') \geq f(x,\xi) - f(x(\xi),\xi)$$

and therefore

$$\begin{aligned} &\max_{\xi \in \mathcal{U}} \max_{\xi' \in \Xi(x(\xi))} \left[f(x,\xi') - f(x(\xi),\xi)\right] \\ =&\max_{\xi \in \mathcal{U}} \left[f(x,\xi) - f(x(\xi),\xi)\right] \end{aligned}$$

which completes the proof. $\square$

Reliability. We can consider the reliable counterpart of an uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ with respect to δ as a strictly robust counterpart of the problem

$$\begin{aligned} P'(\xi) \qquad & \min\ f(x,\xi) \\ & \text{s.t. } F'(x,\xi) \leq 0 \\ & \quad\ x \in \mathcal{X}, \end{aligned}$$

where $F'(x,\xi) := F(x,\xi) - \delta$. Therefore, the same results on the relation between RecFeas and strict robustness can be applied here.

Soft robustness. Setting $r(x,\mathcal{U}(\epsilon)) = \max_{\xi \in \mathcal{U}(\epsilon)} d(x, \mathcal{F}(\xi)$, the RecFeas counterpart of an uncertain optimization problem with an uncertainty set as in soft robustness would be to minimize the objective function $max_{\epsilon > 0} r(x,\mathcal{U}(\epsilon)$. Similar to the strictly robust case, we have that the optimal objective value of the RecFeas counterpart is 0 if and only if there is a softly robust solution.

UFO. We may consider RecFeas and RecOpt as a special case of the UFO approach with finite uncertainty set, where we neglect the budget constraint $f(x) \leq (1+\rho)f^*(\hat{\xi})$. Set

$$\mu(x) = \begin{pmatrix} d(x, \mathcal{F}(\xi^1)) \\ \vdots \\ (x, \mathcal{F}(\xi^N)) \end{pmatrix}$$

then both $\max_{i=1,\dots,N} \mu(x)_i$ and $\sum_{i=1}^N \mu(x)_i$ are possible linearizations of the multi-criteria objective function used in UFO.

3.3.2 Worst-Case Scenarios

In this section we analyze a special case when the concepts of strict and adjustable robustness coincide: The existence of a *worst-case scenario*. For an uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ with

$$\begin{aligned} P(\xi) \quad \min\ & f(x,y,\xi) \\ & F(x,y,\xi) \leq 0 \\ & (x,y) \in X \times Y, \end{aligned}$$

we shortly recall the definition of strict and adjustable robustness as presented in Sections 2.1 and 2.3.

The set of strictly robust solutions is given by

$$\mathrm{SR} = \bigcap_{\xi \in \mathcal{U}} \mathcal{F}(\xi),$$

and for $(x,y) \in \mathrm{SR}$, we evaluate the worst-case objective by

$$z^{\mathrm{SR}}(x,y) = \sup_{\xi \in \mathcal{U}} f(x,y,\xi).$$

The strict robust counterpart to $P(\xi), \xi \in \mathcal{U}$ is given as

$$\inf\{z^{\mathrm{SR}}(x,y) : (x,y) \in \mathrm{SR}\},$$

and we write z^{SR} for short for the optimal objective value, if it exists.

The set of adjustable robust solutions is given by

$$\begin{aligned} \mathrm{aSR} &= \{x \in X : \forall \xi \in \mathcal{U}\ \exists y \in Y \text{ s.t. } (x,y) \in \mathcal{F}(\xi)\} \\ &= \bigcap_{\xi \in \mathcal{U}} Pr_X(\mathcal{F}(\xi)) \end{aligned}$$

where for some set $A \subseteq X \times Y$, $Pr_X(A) = \{x \in X : \exists y \in Y \text{ s.t. } (x,y) \in A\}$ denotes the projection of A on X.

The worst-case objective for some $x \in \text{aSR}$ is given as

$$z^{\text{aSR}}(x) = \sup_{\xi \in \mathcal{U}} \inf_{y:(x,y) \in \mathcal{F}(\xi)} f(x,y,\xi).$$

The adjustable robust counterpart is then given as

$$\inf\{z^{\text{aSR}}(x) : x \in \text{aSR}\}$$

Analogously , we denote by z^{aSR} its optimal value.

In many cases, a strictly robust solution has to be even more buffered than it would be necessary if the scenarios were considered for themselves. This is formally expressed by the value of the robustness gap of the strictly robust solution of an uncertain problem:

Definition 3.45 ([BTN98])**.** Robustness gap
The robustness gap of P is given by

$$gap(P) = z^{\text{SR}} - \sup_{\xi \in \mathcal{U}} \inf_{(x,y) \in \mathcal{F}(\xi)} f(x,y,\xi) \geq 0.$$

In the following, we assume that f is bounded below for all $\xi \in \mathcal{U}$. Note that

$$Pr_X(\text{SR}) \subseteq \text{aSR},$$

i.e., in general we have a larger choice for x if we use the concept of adjustable robustness and hence have a chance to obtain better solution values for the respective robust counterparts. We also cannot get worse, since

$$z^{\text{aSR}}(x) = \sup_{\xi \in \mathcal{U}} \inf_{\tilde{y}:(x,\tilde{y}) \in F(\xi)} f(x,\tilde{y},\xi) \leq \sup_{\xi \in \mathcal{U}} f(x,y,\xi) = z^{\text{SR}}(x,y)$$

for all $(x,y) \in \text{SR}$.

The drawback of adjustable robustness is its in general severe intractability. In [BTGN09], two special cases are presented when the task of solving the adjustable robust counterpart becomes simple: For *fixed-recourse constraints* with an uncertainty set that is given as the convex hull of finitely many scenarios, and for *constraint-wise* uncertainty, where the uncertain data ξ can be split into blocks $[\xi^1, \ldots, \xi^m]$, such that the ith constraint solely depends on ξ^i. In the latter case, the adjustable robust counterpart even becomes a strictly robust problem again.

We now consider further special cases in which strict and adjustable robust solutions coincide, i.e., in which it does not help to keep variables at hold for a *wait-and-see* decision. A crucial role for our results plays the existence of scenarios that are dominant in the following sense:

Definition 3.46. Worst-case scenarios
A scenario $\xi^{WC} \in \mathcal{U}$ is called a global worst-case scenario, if

$$\mathcal{F}(\xi^{WC}) \subseteq \mathcal{F}(\xi) \quad \forall \xi \in \mathcal{U}, \tag{3.10}$$

and

$$f(x, y, \xi^{WC}) = \sup_{\xi \in \mathcal{U}} f(x, y, \xi) \quad \forall (x, y) \in \bigcup_{\xi \in \mathcal{U}} \mathcal{F}(\xi). \tag{3.11}$$

A scenario $\xi^{FWC} \in \mathcal{U}$ satisfying (3.10) is called a feasibility worst-case scenario, while scenarios $\xi^{OWC} \in \mathcal{U}$ satisfying (3.11) are called objective worst-case scenarios.
If a parameter $\xi^{QWC} \in \mathbb{R}^M$ that is not necessarily in $\mathcal{U}$ satisfies (3.10) and

$$\forall i \in \{1, \dots, m\} \; \exists \xi \in \mathcal{U} \; \forall (x, y) \in X \times Y : F_i(x, y, \xi) = F_i(x, y, \xi^{QWC}),$$

we call it a quasi-worst-case scenario.

Note that every global worst-case scenario is also a feasibility worst-case scenario and an objective worst-case scenario, and every feasibility worst-case scenario is also a quasi-worst-case scenario.

We prove the following implications of the existence of worst-case scenarios:

$$\begin{tikzcd}
 & & \exists \xi^{WC} \arrow[dl, Rightarrow, "trivial"'] \arrow[d, Rightarrow, "Th.\ 3.49"] \\
 & \exists \xi^{FWC} \arrow[dl, Rightarrow, "trivial"'] \arrow[d, Rightarrow, "Th.\ 3.48"'] & gap(P) = 0 \arrow[d, Rightarrow, "Th.\ 3.50"] \\
\exists \xi^{QWC} \arrow[d, Rightarrow, "Lem.\ 3.47"'] & Pr_X(\mathbb{SR}) = \mathrm{a}\mathbb{SR} & z^{\mathbb{SR}} = z^{\mathrm{a}\mathbb{SR}} \\
\mathbb{SR} = \mathcal{F}(\xi^{QWC}) & &
\end{tikzcd}$$

We start with noting that feasibility worst-case scenarios already determine the set of strictly robust solutions.

Lemma 3.47. *Let $\xi^{QWC} \in \mathcal{U}$ be a quasi-worst-case scenario. Then*

$$\mathcal{F}(\xi^{QWC}) = \mathbb{SR}.$$

Proof. Since $\mathcal{F}(\xi^{QWC}) \subseteq \mathcal{F}(\xi)$ for all $\xi \in \mathcal{U}$ we directly obtain that

$$\mathcal{F}(\xi^{QWC}) \subseteq \bigcap_{\xi \in \mathcal{U}} \mathcal{F}(\xi) = \mathbb{SR}.$$

We now show that $\mathrm{SR} \subseteq \mathcal{F}(\xi^{QWC})$. There is a set of scenarios $\{\xi^1, \ldots, \xi^m\} \subseteq \mathcal{U}$ such that $F_i(x,y,\xi^i) = F_i(x,y,\xi^{QWC})$ for all $(x,y) \in X \times Y$. Let $(x,y) \in \mathrm{SR}$ be given. As $F_i(x,y,\xi^i) \leq 0$ for all i, we know that $F_i(x,y,\xi^{QWC}) \leq 0$ and therefore $(x,y) \in \mathcal{F}(\xi^{(QWC)})$. Together, $\mathcal{F}(\xi^{QWC}) = \mathrm{SR}$. □

We now can show that for adjustable robustness, we have no larger choice for the solutions for x than in strict robustness, if a feasibility worst-case scenario exists.

Theorem 3.48.
If there exists a feasibility worst-case scenario, it holds that $Pr_X(\mathrm{SR}) = \mathrm{aSR}$.

Proof. Since $\mathrm{SR} = \mathcal{F}(\xi^{FWC})$ due to Lemma 3.47 and $\mathrm{aSR} = \bigcap_{\xi \in \mathcal{U}} Pr_X(\mathcal{F}(\xi))$, it is enough to show that

$$Pr_X(\mathcal{F}(\xi^{FWC})) = \bigcap_{\xi \in \mathcal{U}} Pr_X(\mathcal{F}(\xi)).$$

To this end, we use again that $\mathcal{F}(\xi^{FWC}) \subseteq \mathcal{F}(\xi)$ for all $\xi \in \mathcal{U}$, hence

$$\begin{aligned} & Pr_X(\mathcal{F}(\xi^{FWC})) \subseteq Pr_X(\mathcal{F}(\xi)) \quad \forall \xi \in \mathcal{U} \\ \Rightarrow\ & Pr_X(\mathcal{F}(\xi^{FWC})) \subseteq \bigcap_{\xi \in \mathcal{U}} Pr_X(\mathcal{F}(\xi)). \end{aligned}$$

On the other hand, $\xi^{FWC} \in \mathcal{U}$. This means that

$$Pr_X(\mathcal{F}(\xi^{FWC})) \supseteq \bigcap_{\xi \in \mathcal{U}} Pr_X(\mathcal{F}(\xi))$$

□

Theorem 3.49.
If there exists a global worst-case scenario, the robustness gap is zero.

Proof.
Assume that there is a global worst-case scenario $\xi^{WC} \in \mathcal{U}$. Then,

$$\begin{aligned} \sup_{\xi \in \mathcal{U}} \inf_{(x,y) \in \mathcal{F}(\xi)} f(x,y,\xi) \quad & \geq \quad \inf_{(x,y) \in \mathcal{F}(\xi^{WC})} f(x,y,\xi^{WC}) \\ & \overset{Lem\ 3.47}{=} \inf_{(x,y) \in \mathrm{SR}} f(x,y,\xi^{WC}) \\ & \overset{Eq(3.11)}{=} \inf_{(x,y) \in \mathrm{SR}} \sup_{\xi \in \mathcal{U}} f(x,y,\xi) = z^{SR} \end{aligned}$$

This means that

$$gap(P) = z^{\mathrm{SR}} - \sup_{\xi \in \mathcal{U}} \inf_{(x,y) \in \mathcal{F}(\xi)} f(x,y,\xi) = 0$$

□

Theorem 3.50.
If the robustness gap is zero, then $z^{\mathrm{SR}} = z^{\mathrm{aSR}}$.

Proof.
To show that $z^{\mathrm{SR}} \leq z^{\mathrm{aSR}}$, let $x^* \in \mathrm{aSR}$ be such that $z^{\mathrm{aSR}} \geq z^{\mathrm{aSR}}(x^*) - \varepsilon$ for some $\varepsilon > 0$. Because of $gap(P) = 0$, we have

$$\begin{aligned} z^{\mathrm{SR}} &= \sup_{\xi \in \mathcal{U}} \inf_{(x,y) \in \mathcal{F}(\xi)} f(x, y, \xi) \\ &\leq \sup_{\xi \in \mathcal{U}} \inf_{y:(x^*,y) \in \mathcal{F}(\xi)} f(x^*, y, \xi) \\ &= z^{\mathrm{aSR}}(x^*) \leq z^{\mathrm{aSR}} + \varepsilon \end{aligned}$$

On the other hand, we already know that $z^{\mathrm{SR}} \geq z^{\mathrm{aSR}}$. For $\varepsilon \to 0$, we therefore have $z^{\mathrm{SR}} = z^{\mathrm{aSR}}$. □

We see that the existence of a global worst-case scenario yields the strongest implications. Experience from practical applications of robustness suggests that this condition is indeed satisfied for numerous problem classes, in particular for interval-wise uncertainty sets, which are in general not *constraint-wise*. Using these results, practitioners are given tools to identify cases in which not both concepts have to be tested beforehand, and the higher computational effort for solving the adjustable robust counterpart may be avoided.

4 Continuous Problem Applications

Following the methodology outlined in Section 1.3, we proceed to analyze the performance of robustness concepts in practical applications. We begin with linear programs from the well-known NetLib benchmark set [DG12] in Section 4.1, where we compare RecOpt solutions to the nominal and strictly robust solutions, and compare the quality of solutions generated by the iterative Algorithm 1 and the sampling Algorithm 2 for RecFeas. We then consider aperiodic timetabling instances in Section 4.2, where we can compare the concept of RecOpt to strict, light and recovery robustness, as well as the nominal and the equally buffered solution.

4.1 Linear Programming

In this section we consider the performance of RecOpt and RecFeas on linear programming instances. Using a benchmark set, we cannot assume knowledge about a certain problem structure, but apply a generic problem reformulation instead. We begin with RecOpt in Section 4.1.1, and turn to RecFeas in Section 4.1.2.

4.1.1 RecOpt

In this section we present numerical results for the performance of RecOpt by randomly disturbed problems taken from the NetLib [DG12] library. We begin with describing the experimental environment and setup, and then discuss the results gained.

Environment. The RecOpt experiments described here were performed on an AMD Quad-Core Opteron 2354 with 4 cores at 2.2 GHz and 33 GB RAM, running Ubuntu. All linear programs were solved using Gurobi 3.0 [Gur10].

Setup. We have to decide on how to measure distances between two solutions, representing the “costs” for updating a solution. This measure depends on the problem under consideration. In our experiments, we decided to use the Manhattan distance l_1 as a canonic choice for testing purposes.

Our goal is to compare our robust solutions according to RecOpt with the *nominal solution*, i.e., the solution for the undisturbed scenario, and with the strictly robust solution. To increase the likeliness of the existence of a strictly robust solution, we modified the constraints of the NetLib problems using the following scheme.

1. Choose an LP and consider it as nominal instance.
2. Modify every "="-constraint to a "$\leq$"-constraint.
3. Add the constraint that all variables are positive.
4. Check if there is still a finite optimum. If not, neglect this LP.
5. Create an index set $J = \{j_1, \dots, j_N\}$ by selecting a ratio of $0 \leq p \leq 1$ of the indices of the coefficient matrix A, restricted to its non-zero entries.
6. The uncertainty set for this instance is then given by

$$\mathcal{U} = \left\{ \hat{A} : (1-q)a_j \leq \hat{a}_j \leq (1+q)a_j \ \forall j \in J, \ \hat{a}_j = a_j \ \forall j \notin J \right\},$$

   where q represents the allowed deviation and $\hat{A}$ the coefficient matrix.
7. Calculate
   a) the nominal solution,
   b) the strictly robust solution,
   c) and apply Algorithm 3 for
      i. the center with respect to the Manhattan norm l_1,
      ii. the center with respect to l_1 with additional *nominal feasibility*, i.e., with feasibility to the nominal scenario,
      iii. and the centroid.

Our modification in Step 2 increases the probability that a strictly robust solution exists, while the modification in Step 3 helps us to calculate the strictly robust solution: If all variables are greater or equal to zero, the worst case scenario and hence the strictly robust solution can be easily calculated.

To find the l_1 centers, we used linear programming formulations. Furthermore, to avoid exceptional long computation times, we added a time limit of 180 seconds to each LP.

Of the 94 problems from NetLib, 22 had to be neglected by Step 4. Of the remaining 72 problems, 52 had a strictly robust solution and all solutions were found within the time limit. For each of these 52 instances we calculated the five solutions mentioned in Step 7.

For the first set of experiments, we set $p = 0.4$ and $q = 0.4$, meaning that 40% of the non-zero entries can deviate by up to 40%. An interpretation would depend on the respective problem the LP models – say, bad weather conditions that disturb routes in a routing problem. For the second set, we set $p = 0.2$ and $q = 0.8$. In the former example, this could be caused by traffic jam on some routes.

	$p = 0.4,\ q = 0.4$			$p = 0.2,\ q = 0.8$		
	feasibility	objective	radius	feasibility	objective	radius
nominal	0.27	0.00	0.00	0.28	0.00	0.00
l_1 center	0.28	2.18	-0.33	0.26	1.89	-0.40
l_1 center w.n.f.	0.30	2.96	-0.32	0.31	2.55	-0.37
centroid	0.29	-0.98	-0.19	0.29	0.20	-0.14
strict robustness	1.00	5.92	2.37	1.00	95.30	1.92

Table 4.1 Average feasibility, objective value ratio and recovery radius ratio of the 52 NetLib instances. "w.n.f." abbreviates "with nominal feasibility".

Results. Table 4.1 gives an overview on the evaluation of these solutions for the two choices of the parameters p, q.

For the column "feasibility" we evaluated every solution x^* as follows: We randomly picked 1,000 scenarios from the uncertainty set and counted for how many of them x^* is feasible; the average feasibility over all scenarios is shown. As expected, the strictly robust solution is always feasible. The l_1 centers and the centroid nearly always have slightly better feasibility than the nominal solution.

The column "objective" represents the relative change of the objective value. For a solution with objective value x_{sol}, we calculated

$$\text{objective} = \frac{x_{\text{sol}} - x_{\text{nom}}}{|x_{\text{nom}}|},$$

where x_{nom} denotes the objective value of the nominal optimum. This means that higher values represent more costly solutions. The table shows that the costs of the strictly robust solutions dramatically increase, while the solutions calculated by RecOpt have only slightly higher costs than the nominal solution, or even smaller ones – which is possible, since a solution to RecOpt is not necessarily feasible for the nominal scenario.

The values of the column "radius" represent the approximate maximum recovery costs. As it was the case in the objective column, we calculated the radius in relation to the recovery radius of the nominal solution, i.e.,

$$\text{radius} = \frac{r_{\text{sol}} - r_{\text{nom}}}{r_{\text{nom}}},$$

where r_{nom} denotes the recovery costs for the nominal solution, and r_{sol} the recovery costs for the solution under consideration. It turns out that the recovery radius for the strictly robust solutions is even higher than for the nominal solutions, while RecOpt yields solutions with smaller recovery costs.

Summarizing, Table 4.1 shows that RecOpt is a good choice for robust solutions that do not need to be feasible for every scenario and combines the advantages of good objective values with low repair costs.

4.1.2 RecFeas

We now evaluate the performance of the Algorithms 1 and 2 for RecFeas on the NetLib instances. As before, we begin with describing the experimental environment and setup, and then discuss the computational results.

Environment. These experiments were conducted on an Intel Xeon X5650 with 6 cores at 2.66 GHz and 99 GB RAM, running Ubuntu. Linear programs were solved using Gurobi 5.0 [Gur12].

Setup. We considered the smallest 20 instances of NetLib, and dropped those where $x = 0$ is infeasible due to variable bounds. For the remaining 14 instances, we proceeded as follows: We generated uncertainty sets $\mathcal{U}$ of the size 10, 20, 30, 40, and 50; 10 for each size. Scenarios were generated by multiplying a random non-zero coefficient by a random number from the interval $[0.75, 1.25]$. Infeasible scenarios are dropped until the required number is generated. We then calculated the exact solution of the RecFeas counterpart without nominal feasibility, and with respect to the center l_1 objective function. Furthermore, for each of the 50 uncertainty sets, we sampled $|\mathcal{U}|/2$ scenarios at random and solved the respective smaller RecFeas problem. Finally, we used the iterative algorithm from Section 3.1.5.1 with two different termination criteria: Once if the difference between $r^{(k)}$ and $r^{(k-1)}$ is smaller than $10^{-3} \cdot r^{(0)}$ (Iterative-A), and once if it is smaller than $10^{-4} \cdot r^{(0)}$ (Iterative-B). For every solution, we measured the running time and the recovery distance.

Results. The average computation time is shown in Table 4.2. While the iterative algorithm is slower for problems with a small uncertainty size, it scales better than the sampling approach for increasing problem size, and is the fastest algorithm for large instances. The number of iterations as presented in Table 4.3 increases with the uncertainty size.

$\lvert\mathcal{U}\rvert$	Exact	Sample	Iterative-A	Iterative-B
10	1.23	0.21	1.26	1.55
20	11.41	1.38	2.89	3.80
30	33.96	3.99	3.66	4.36
40	68.71	8.09	4.51	5.66
50	92.51	13.19	6.16	8.41

Table 4.2 Average computation time in s.

The comparison of objective values, however, is more ambiguous. In Table 4.4 we show the arithmetic mean of the recovery radii, and a normalized mean. Normalized values were calculated a linear function that maps the best solution of an instance to the value 0, and the worst solution to the value 1. When all solutions have the same objective value, they are all normalized to 0.

$\lvert\mathcal{U}\rvert$	Iterative-A	Iterative-B
10	2.21	2.67
20	2.66	3.41
30	2.62	3.25
40	2.85	3.54
50	2.99	4.68

Table 4.3 Average number of iterations.

	Exact		Sample		Iterative-A		Iterative-B	
$\lvert\mathcal{U}\rvert$	Mean	Norm.	Mean	Norm.	Mean	Norm.	Mean	Norm.
10	10.68	0.00	36.42	0.43	35.80	0.47	32.80	0.45
20	13.89	0.00	38.81	0.40	60.32	0.51	51.01	0.48
30	22.22	0.00	68.96	0.45	66.27	0.44	60.55	0.41
40	24.55	0.00	84.21	0.44	117.34	0.49	113.08	0.47
50	32.26	0.00	155.35	0.46	137.57	0.45	131.02	0.44

Table 4.4 Average recovery radius.

We note that the average optimal recovery radius increases for increasing uncertainty set size, as we may expect. A comparison of the sampling approach and the iterative approach does not indicate that one outperforms the other; in fact, they generate solutions of about the same quality. To analyze their behavior in more detail, we compare the rank between both algorithms, which we define as follows: If a solution of an algorithm has a strictly better objective value than the solution of the other, it is granted one point. We then calculate the average number of points for every uncertainty set, and present the results in Table 4.5. For instances with $|\mathcal{U}| \geq 30$, solutions of the iterative approach are on average better than solutions of the sampling approach. We remark that although the rank values are equal for $|\mathcal{U}| = 30, 40, 50$, the instances where one approach dominates the other are different.

$\lvert\mathcal{U}\rvert$	Sample	Iterative-B
10	0.29	0.29
20	0.43	0.14
30	0.21	0.36
40	0.21	0.36
50	0.21	0.36

Table 4.5 Rank.

Finally, we compare all objective values of the sampling and the iterative approach over all instances against each other in Figure 4.1. A data point that is *below* the identity function, which is plotted as the straight line, means that the solution from the iterative approach has a *better* objective value than its sampling counterpart.

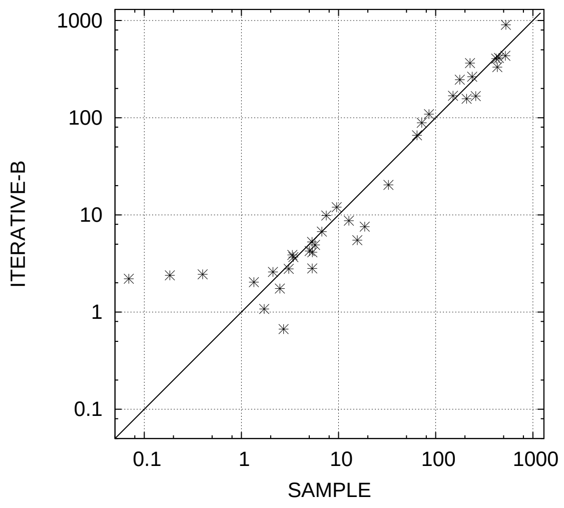


Figure 4.1 Objective values for Sample and Iterative-B.

We find that although most data points are below the line, the iterative approach performed significantly worse than the sampling approach on some instances with very small recovery radius. We can explain this behavior from the setting of the experiment: The iterative algorithm terminates when the objective value difference between two iterations is below $10^{-4} \cdot r^{(0)}$. When the optimal solution has an objective value that is close to 0, and $x^{(0)} = 0$ has a large recovery radius, the algorithm terminates too early.

4.1.3 Conclusion

We used instances from the LP benchmark set NetLib to compare solutions generated by the concept RecOpt in different variations to the nominal and the strictly robust solution. We remark that RecOpt and strict robustness aim at different objectives: While the former minimizes the recovery distance to the respective optimal solutions, the latter minimizes the objective value under maximum feasibility. Therefore, each method excels in its own domain. The results show the *trade-off* between these different objectives, which indicates a good performance of our approach.

Furthermore, for 33 of the 42 omitted NetLib instances, no strictly robust solution exists. Hence, in particular for these instances the solutions calculated by RecOpt are the better choice.

For the experiments concerning RecFeas, we compared the quality of heuristic algorithms instead of concepts. We used the sampling and the iterative method to calculate RecFeas solutions, and compared them to the exact robust solutions. We found that even though they yield about the same level of robustness, the iterative approach performs better when we ignore solutions with small optimal recovery radius, which were hard to solve due to the termination criterion that was determined based on the starting solution. Furthermore, running times of the iterative approach scale better for an increasing uncertainty set size.

4.2 Aperiodic Timetabling

4.2.1 Introduction

We now consider a problem that has received considerable attention in recent robust optimization literature – see, e.g., [DSNP09, FM09, LLMS09]. The *aperiodic timetabling problem* is one of significant importance in real-world applications where it is needed to create timetables that stay "good" under the unavoidable small disturbances of daily railway operations. Robust solutions usually lead to high buffer times, which in turn yield high traveling times and thus unattractive timetables.

Contribution. In this section we compare some of the most prominent robustness concepts from the literature, as well as the RecOpt approach, for the timetabling problem on a real-world instance numerically.

We analyze two different macroscopic types of uncertainty: One that allows small delays on all edges, and one that allows heavy delays on a restricted set of edges, and show empirically that the structure of these determine which robustness concept fits best.

Overview. In Section 4.2.2, we formally introduce the aperiodic timetabling problem, and discuss the application of robustness concepts in Section 4.2.3. In Section 4.2.4, we describe an experimental study that compares these concepts on a real-world instance.

4.2.2 Problem Formulation

The problem we consider is the following: Let an *event-activity-network* (EAN) be given, that is, a directed graph $G = (\mathcal{E}, \mathcal{A})$ consisting of departure and arrival *events*

$$\mathcal{E} = \mathcal{E}^{\text{arr}} \cup \mathcal{E}^{\text{dep}}$$

and waiting, driving, changing and headway *activities*

$$\mathcal{A} = \mathcal{A}^{\text{wait}} \cup \mathcal{A}^{\text{drive}} \cup \mathcal{A}^{\text{change}} \cup \mathcal{A}^{\text{head}}.$$

Driving activities $\mathcal{A}^{\text{drive}} \subseteq \mathcal{E}^{\text{dep}} \times \mathcal{E}^{\text{arr}}$ represent traveling from one station to another, while waiting activities $\mathcal{A}^{\text{wait}} \subseteq \mathcal{E}^{\text{arr}} \times \mathcal{E}^{\text{dep}}$ represent staying of a train at a station while passengers board and deboard. Changing activities $\mathcal{A}^{\text{change}} \subseteq \mathcal{E}^{\text{arr}} \times \mathcal{E}^{\text{dep}}$ model passengers who plan to change from one train to another at the same station, while headways $\mathcal{A}^{\text{head}} \subseteq \mathcal{E}^{\text{dep}} \times \mathcal{E}^{\text{dep}}$ are introduced to model safety distances between trains sharing the same infrastructure.

Figure 4.2 gives an example for the general structure of such a network. In station A, passengers would like to change from a train of line 1 into a train of line 2 and vice versa. In station B, there are passengers who change from line 2 to line 3.

Assigned to each of these activities $(i, j) \in \mathcal{A}$ is a minimal duration $\hat{l}_{ij} \in \mathbb{N}$ representing the technically possible lower time bound for an activity to take place, and a

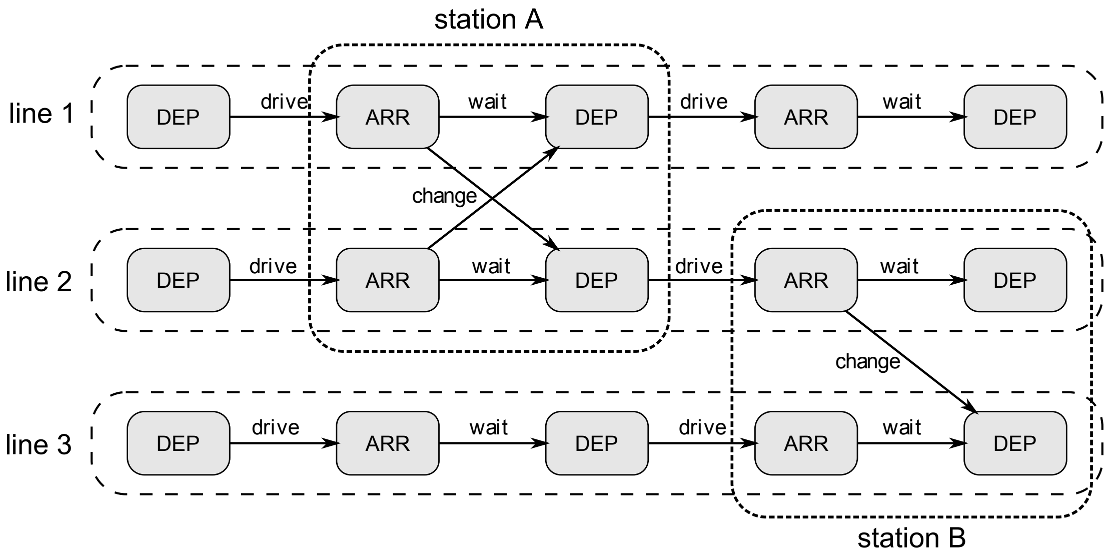


Figure 4.2 Detail of an Event-Activity-Network.

number w_{ij} of passengers using activity $(i,j) \in \mathcal{A}$. The task is to find node potentials $\pi_i \in \mathbb{R}$ for all $i \in \mathcal{E}$, such that the sum of passenger traveling times $w_{ij}(\pi_j - \pi_i)$ over all activities $(i,j) \in \mathcal{A}$ is minimized for given passenger weights w_{ij} under the time restrictions $\pi_j - \pi_i \geq \hat{l}_{ij}$ for each activity $(i,j) \in \mathcal{A}$. Its well-known mathematical formulation is

$$\text{(TT)} \quad \min \sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i) \tag{4.1}$$

$$\text{s.t.} \quad \pi_j - \pi_i \geq \hat{l}_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{4.2}$$

$$\pi_i \geq 0 \quad \forall i \in \mathcal{E}. \tag{4.3}$$

We may also extend this problem by adding upper bounds u_{ij} on activity durations $(i,j) \in \mathcal{A}$. We will sometimes simply write $\hat{l} = (\hat{l}_{ij})_{(i,j)\in\mathcal{A}}$ as the vector of all lower bounds, and similarly $\pi = (\pi_i)_{i\in\mathcal{E}}$ as the vector of node potentials, for any given edge- and node order. Note that the time restrictions form a totally unimodular matrix, i.e. even though real node potentials might be considered as unrealistic in railway operations, we will always find an integer optimal solution. Furthermore, (TT) is feasible for all possible activity durations $\hat{l} \geq 0$ if the network does not contain any directed cycle with positive length, infeasible otherwise.

4.2.3 Robustness Concepts

In order to hedge (TT) against delays in operation, we have to model the possible disturbances first. Which (source) disturbances occur is in practice not known beforehand, since this depends on exterior influences like weather conditions or technical failures. Hence the activity durations are uncertain. Here we assume that the passenger distribution $w = (w_{ij})_{(i,j)\in\mathcal{A}}$, i.e., the number of passengers using each activity, is known.

The first type of uncertainty we consider is one of *uniform deviation.* Imagine, for example, bad weather conditions that *slightly* delay all trains on track equally. We model this behavior with the following set of scenarios depending on $s \in \mathbb{R}^+$, where s controls the level of uncertainty:

$$\begin{aligned}\mathcal{U}_1(s) := \{l : \hat{l}_{ij} \le l_{ij} \le (1+s)\hat{l}_{ij}\ \forall (i,j) \in \mathcal{A}^{\text{drive}} \cup \mathcal{A}^{\text{wait}},\\ l_{ij} = \hat{l}_{ij}\ \forall (i,j) \in \mathcal{A}^{\text{change}} \cup \mathcal{A}^{\text{head}}\}\end{aligned}$$

The second type of uncertainty we analyze models the situation that only a restricted number of activities may be delayed at the same time, but *heavier.* This may be the case when good weather conditions hold but single trains are delayed by blocked tracks or technical failures. For $k \ge 1$, we define

$$\begin{aligned}\mathcal{U}_2(k,s) := \{l : \hat{l}_{ij} \le l_{ij} \le (1+s)\hat{l}_{ij}\ \forall (i,j) \in D \subseteq \mathcal{A}^{\text{drive}} \cup \mathcal{A}^{\text{wait}},\ |D| = k,\\ l_{ij} = \hat{l}_{ij}\ \forall (i,j) \in \mathcal{A} \setminus D\}\end{aligned}$$

Using $\mathcal{U}_2$ we assume that not all, but at most k lower bounds change to their worst values in the same scenario, which can be interpreted in the sense of Bertsimas and Sim [BS04] in the dual problem.

We may consider problem (TT) as an uncertain optimization problem

$$\min\{f(\pi) : F(\pi, l) \le 0\}$$

w.r.t l, where the objective function $f(\pi) = \sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i)$ does not depend on l and the constraints are given as $F(\pi, l) = (l - A^t\pi)$, where A is the node-arc incident matrix of G, that is, $a_{ie} = 1$, if $e = (j,i)$ for a $j \in \mathcal{E}$, $a_{ie} = -1$, if $e = (i,j)$ for a $j \in \mathcal{E}$, and $a_{ie} = 0$ else, and $l \in \mathbb{R}^{|\mathcal{A}|}$ contains the minimum activity durations. Note that for $\mathcal{U}_1$ and $\mathcal{U}_2$, we have $M = m$. We now show how recent robustness concepts can be applied to the timetabling problem.

Strict Robustness. The concept requires feasibility of a robust solution under all possible scenarios, i.e. that $F(x, \xi) \le 0$ for all $\xi \in \mathcal{U}$. For (TT) we obtain

$$\text{(S-TT)} \quad \min \sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i) \tag{4.4}$$

$$\text{s.t.} \quad \pi_j - \pi_i \ge l_{ij} \quad \forall (i,j) \in \mathcal{A} \text{ and } \forall l \in \mathcal{U} \tag{4.5}$$

$$\pi \ge 0. \tag{4.6}$$

(S-TT) is called the strict robust counterpart of (TT). In general, this leads to infinitely many constraints, depending on the choice of $\mathcal{U}$. It is shown in [BTN98] that if

$\mathcal{U} = \mathrm{conv}\{\xi^1, \dots, \xi^N\}$, and $F(x, \cdot), f(x, \cdot)$ are quasiconvex in ξ, then the strict robust counterpart is equivalent to a program where the constraints only have to be satisfied for $\xi^1, \dots, \xi^N$. This is evidently the case for (TT) with $\mathcal{U}_1$ as defined above. Omitting redundant constraints we hence gain the following strict robust formulation for $\mathcal{U}_1$:

$$\text{(S-TT)} \quad \min \sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i) \tag{4.7}$$

$$\text{s.t.} \quad \pi_j - \pi_i \geq (1+s)\hat{l}_{ij} \quad \forall\ (i,j) \in \mathcal{A} \tag{4.8}$$

$$\pi \geq 0 \tag{4.9}$$

In case of $\mathcal{U}_2$, the same result holds due to the fact that all but the listed constraints (4.8) become dominated by other scenarios – the scenario $(1+s)\hat{l}$ is a quasi-worst-case in the sense of Definition 3.46. Note that the guaranteed feasibility comes at a high price, as the maximum buffer is put on every edge even though only a few may become delayed.

Light Robustness. As before, let m be the number of constraints. Variables γ_i are introduced for each constraint $i = 1, \dots, m$ of the nominal problem that measure the degree of relaxation needed for strict robustness. The goal is to minimize the sum of these γ_i while guaranteeing a certain quality of the solution. Let the nominal scenario be denoted by $\hat{\xi} \in \mathcal{U}$ and let $z^* > 0$ be the optimal objective value of the nominal problem. Then, for a given δ, the light robustness approach to the timetabling problem (TT) with uncertainty $\mathcal{U}_1$ corresponds the following program:

$$\text{(L-TT)} \quad \min \sum \gamma_{ij} \tag{4.10}$$

$$\text{s.t.} \quad \sum w_{ij}(\pi_j - \pi_i) \leq (1+\delta)z^* \tag{4.11}$$

$$\pi_j - \pi_i \geq \hat{l}_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{4.12}$$

$$\pi_j - \pi_i \geq (1+s)\hat{l}_{ij} - \gamma_{ij} \quad \forall (i,j) \in \mathcal{A} \tag{4.13}$$

$$\gamma, \pi \geq 0 \tag{4.14}$$

where we dropped dominated constraints. Note that $\hat{l}$ is used as the nominal scenario. Constraint (4.13) dominates all other scenarios $l \in \mathcal{U}$, and is therefore sufficient – again, as $(1+s)\hat{l}$ is a quasi-worst-case. Also here $\mathcal{U}_2$ yields the same formulation as we can again drop dominated constraints.

Recoverable Robustness. The concept of recoverable robustness was introduced by Liebchen et al. in [LLMS09] and by Cicerone et al. in [CDS$^+$09c, DDNP11, CDS$^+$09a], both groups also proposing applications to timetabling. The basic idea is to find a robust solution that can be "repaired", i.e., made feasible by delaying events, with low costs as soon as the real scenario becomes known. In both papers [LLMS09, CSSS08], the sum of all arrival delays of the passengers and the maximum delay of each arrival event are restricted by budget parameters λ_1 or λ_2. As these budget parameters might be difficult to estimate in advance, they are regarded as variables in [LLMS09] and become part of the objective function with according weights, say g_1 and g_2. Denoting by $\tilde{w}_i$, $i \in \mathcal{E}^{\text{arr}}$, the number of passengers de-boarding at event i, and assuming a finite set of scenarios $\mathcal{U}$, [LLMS09] suggest the following program:

$$\begin{aligned}
\text{(R-TT)} \quad \min \quad & \sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i) + g_1\lambda_1 + g_2\lambda_2 && (4.15)\\
\text{s.t.} \quad & \pi_j - \pi_i \geq \hat{l}_{ij} \quad \forall (i,j) \in \mathcal{A} && (4.16)\\
& \pi^l_j - \pi^l_i \geq l_{ij} \quad \forall l \in \mathcal{U},\ \forall\ (i,j) \in \mathcal{A} && (4.17)\\
& \pi^l_i \geq \pi_i \quad \forall l \in \mathcal{U},\ \forall i \in \mathcal{E}^{\text{dep}} && (4.18)\\
& \sum_{i\in\mathcal{E}^{\text{arr}}} \tilde{w}_i(\pi^l_i - \pi_i) \leq \lambda_1 \quad \forall l \in \mathcal{U} && (4.19)\\
& \pi^l_i - \pi_i \leq \lambda_2 \quad \forall l \in \mathcal{U},\ \forall i \in \mathcal{E}^{\text{arr}} && (4.20)\\
& \lambda_1, \lambda_2, \pi^l, \pi \geq 0 && (4.21)
\end{aligned}$$

Regarding the number of variables, note that for each scenario a timetabling problem has to be solved. The concept was originally designed for an uncertainty of type $\mathcal{U}_2$, meaning that $\binom{|\mathcal{A}|}{k} + |\mathcal{A}| + 2$ variables need to be created. For $k > 1$ this becomes quickly intractable. For $k = 1$ exactly one activity is delayed per scenario and we may write $\mathcal{U} \cong \mathcal{A}^{\text{wait}} \cup \mathcal{A}^{\text{drive}}$ for short. The authors present a possibility to reformulate the recovery robust timetabling problem in a more compact way by setting $g_2 = 0$ and introducing a fixed recovery budget D instead of using λ_1. For every scenario e variables $y^e = \pi^e - \pi \in \mathbb{R}^{|\mathcal{E}|}$ are needed. Using slack variables f one obtains

$$\begin{aligned}
\text{(R2-TT)} \quad \min_{\pi,f} \quad & \sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i) && (4.22)\\
\text{s.t.} \quad & \pi_j - \pi_i - f_{ij} = \hat{l}_{ij} \quad \forall (i,j) \in \mathcal{A} && (4.23)\\
& f_{ij} + y^e_j - y^e_i \geq s\chi_{ij}(e) \quad \forall (i,j) \in \mathcal{A},\ \forall e \in \mathcal{A}^{\text{wait}} \cup \mathcal{A}^{\text{drive}} && (4.24)\\
& D \geq \|y^e\|_1 \quad \forall e \in \mathcal{A}^{\text{wait}} \cup \mathcal{A}^{\text{drive}} && (4.25)\\
& f, y^e, \pi \geq 0, && (4.26)
\end{aligned}$$

where $\chi_{ij}(e) = 1$ if $e = (i,j)$ and zero else. In this formulation we changed the weights $\tilde{w}$ to be 1 for all nodes for better comparability with other models; however, also other weights may be considered.

For the uncertainty $\mathcal{U}_1$ we obtain a different formulation. Here it is sufficient to find a recovery solution π^{worst} for only the worst-case scenario in which all activity durations take their worst values. Hence, by setting $\tilde{w}_i = 1$ for all $i \in \mathcal{E}$ again, (R-TT) simplifies to

$$\text{(R1-TT)} \quad \min \sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i) + g_1\lambda_1 + g_2\lambda_2 \qquad (4.27)$$

$$
\begin{aligned}
\text{s.t.}\quad & \pi_j - \pi_i \geq \hat{l}_{ij} \quad \forall (i,j) \in \mathcal{A} && (4.28)\\
& \pi_j^{worst} - \pi_i^{worst} \geq (1+s)\hat{l}_{ij} \quad \forall (i,j) \in \mathcal{A} && (4.29)\\
& \pi_i^{worst} \geq \pi_i \quad \forall\, i \in \mathcal{E} && (4.30)\\
& \|\pi^{worst} - \pi\|_1 \leq \lambda_1 && (4.31)\\
& \pi_i^{worst} - \pi_i \leq \lambda_2 \quad \forall i \in \mathcal{E} && (4.32)\\
& \lambda_1, \lambda_2, \hat{\pi}, \pi \geq 0. && (4.33)
\end{aligned}
$$

RecOpt. We now consider the approach that aims to minimize the expected or the maximum repair costs to an *optimal* solution of a scenario, measured in terms of a distance function. Recall that the robust counterpart of this approach is in general notation given as

$$
\begin{aligned}
\text{(RecOpt)}\quad & \min_x \sup_{\xi \in \mathcal{U}} d(x, x^\xi)\\
\text{s.t.}\quad & x^\xi \text{ is an optimal solution to } (\mathrm{P}(\xi)),
\end{aligned}
$$

where $d(x, x^\xi)$ represents the recovery costs needed to update a timetable x to another timetable x^ξ. Instead of the supremum also the average recovery costs may be considered. Since we recover not to a feasible, but to an *optimal* solution x^ξ, a strictly robust solution has no recovery costs in (R-TT), but especially in timetabling will usually have high recovery costs in the sense of (RecOpt). Recovery to optimality may also mean to let events take place *earlier* which is reasonable when a timetable needs not be adapted to the scenario during the operational phase, but the scenario is known some time before – like in the case of track maintenance or exceptional weather forecasts.

Instead of solving (RecOpt) to optimality, we use the Algorithm 3 in which we create a number of scenarios ξ, solve them separately, and find the robust solution by solving a location problem in which the given facilities are the respective optimal solutions of the instances $(\mathrm{P}(\xi))$. Thus we apply the following algorithm to the timetabling problem:

Algorithm 4 (RecOpt-TT)

```
Require: A robust aperiodic timetabling instance (TT), a sample size $N \in \mathbb{N}$ and a
    distance measure $d : \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{E}|} \to \mathbb{R}$.
1: Choose a subset $S \subseteq \mathcal{U}$ of $N$ elements at random.
2: Create vectors $\pi^l \in \mathbb{R}^{|\mathcal{E}|}$ by solving TT($l$) for each $l \in S$.
3: Find a vector $\pi \in \mathbb{R}^{|\mathcal{E}|}$ by minimizing the sum/maximum of distances $d(\pi, \pi^l)$ for
    all $l \in S$.
4: return A robust solution $\pi$.
```

This algorithm is generically applicable to any other robust problem, but has to be specified to its respective needs. In particular, we have to determine which distance measure represents the recovery costs best, how many scenarios should be chosen, and how they should be created.

Note that this heuristic is easily applicable whenever a method for solving the nominal problem is available. Only a generic location problem solver has to be used, while existing algorithms need not be changed.

For our numerical evaluation we used the same recovery costs as in (R1-TT) and (R2-TT), which is the $\|\cdot\|_1$-distance, either in combination with a sum or a maximum, and we added the squared Euclidean distance as third alternative. The resulting combinations are shown in Table 4.6.

Distance (recovery costs)	sum/max	Name	Calculation
$d_1(x,y) = \|x-y\|_1$	sum	d_1 median	$\text{argmin}_\pi \sum_{l\in S}\sum_{i\in\mathcal{E}} \lvert\pi_i - \pi_i^l\rvert$
$d_1(x,y) = \|x-y\|_1$	max	d_1 center	$\text{argmin}_\pi \max_{l\in S}\sum_{i\in\mathcal{E}} \lvert\pi_i - \pi_i^l\rvert$
$d_2^2(x,y) = \|x-y\|_2^2$	sum	centroid	$\frac{1}{\lvert S\rvert}\sum_{l\in S}\pi^l$

Table 4.6 Evaluated distance - sum/max combinations.

Note that we are free to add further restrictions to the location of π. Since a nominal infeasible timetable would not be of practical use, we additionally impose nominal feasibility constraints and solve restricted location problems. We remark that there can be an optimal d_1 center or median for the timetabling problem, that is not feasible for the nominal scenario. In contrast to this, the centroid is always feasible.

Lemma 4.1. *Let a (TT) instance with an uncertainty set $\mathcal{U}_1$ or $\mathcal{U}_2$ be given, and let $d = d_2^2$. Then the robust solution calculated by the sum version of (RecOpt-TT) is nominal feasible for any finite set $S \subseteq \mathcal{U}$.*

Proof. This follows directly from Lemma 3.36. □

Concerning the amount of sampled scenarios N, we tested numerically how many scenarios were needed for a convergence of solutions. This was already the case for less than 100 instances on the instances described in Section 4.2.4.

Note that the uncertainty $\mathcal{U}_1$ is a polyhedral set with a finite number of extreme points, while $\mathcal{U}_2$ is not convex for fixed k and s. By introducing slack variables f as in (R2-TT), we may rewrite the constraints $\pi_j - \pi_i \geq l_{ij}$ of the timetabling problem to $\pi_j - \pi_i - f_{ij} = b_{ij}$. We hence gain the following corollary to Lemma 3.38:

Corollary 4.2. *Let a (TT) instance with an uncertainty set $\mathcal{U} = \text{conv}\{l^1,\ldots,l^N\}$ be given. Let the optimal solution be unique in every scenario. Assume that there is a basis B that is optimal for each scenario $l \in \mathcal{U}$. Then the d_1 center with respect to the solutions $x^{l^1},\ldots x^{l^N}$ solves (RecOpt) applied to the timetabling problem optimally, i.e. the choice $S = \{l^1,\ldots,l^N\}$ in Step 1 of (RecOpt-TT) leads to an exact optimal solution.*

Applying the theory from RecOpt to the timetabling problem, we may conclude:

Corollary 4.3. *Let a (TT) instance with uncertainty set $\mathcal{U}_1$, be given. Let $l^* := (1+s/2)\hat{l}$ and assume that there is a basis that is optimal for $TT(\hat{l})$ and $TT((1+s)l)$. Then any optimal solution to $TT(l^*)$ solves (RecOpt) for the timetabling problem for every distance d that stems from a norm.*

4.2.4 Numerical Studies

Environment. All computations were carried out on a Quad-Core AMD Opteron Processor running at 2.2 GHz with 33 GB RAM using *Gurobi* 3.0 [Gur10] under Ubuntu.

Problem instance and parameters. The instance was created using the *LinTim* toolbox [GSS] for optimization in public transportation based on an intercity train network with the size of the German IC/ICE railway system. The time horizon under consideration consists of the eight-hour service period from 8 a.m. to 4 p.m., resulting in an EAN with 379 activities and 377 events.

We set for (R1-TT) $g_1 = 50$, $g_2 = 10,000$ to gain a solution which is a good compromise in robustness as well as in objective value. The budget D for (R2-TT) was set to 2000. The budget δ for light robustness was set to 0.1, meaning that the objective value of the lightly robust solution is allowed to deviate up to 10 percent with respect to the nominal optimality. Furthermore, we tested a simple *uniformly buffered* solution by multiplying all node potentials of the nominal optimum with 1.06, which increased all activity durations by 6 percent, a method which is often applied in practice.

Concerning the choice of $S \subseteq \mathcal{U}$ for (RecOpt-TT), we tested two versions. In the first, we restricted the choice to extreme points of $\mathcal{U}$, in the second we chose uniformly over the whole uncertainty set. Our results showed a better performance of the latter approach regarding recovery costs to feasibility and optimality for $\mathcal{U}_1$, but a slightly better performance for the extreme points approach for $\mathcal{U}_2$. For the following evaluations we present the (RecOpt-TT) solutions under this respective scenario choice: For $\mathcal{U}_1$, the scenarios were chosen uniformly over the whole uncertainty set, for $\mathcal{U}_2$ only from the extreme points.

Setup. We tested the $\mathcal{U}_1$ algorithms for $s = 0, \ldots, 0.3$ and the algorithms for $\mathcal{U}_2$ with $s = 0, \ldots, 1$ and $k = 1$. For each algorithm and iteration the following values were measured:

- Objective value: $\sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i)$
- Average relative buffer: $1/|\mathcal{A}|(\sum_{(i,j)\in\mathcal{A}}(\pi_j - \pi_i)/l_{ij}) - 1$
- Average costs, when recovering to feasibility: A large number of scenarios l^q, $q = 1, \ldots, Q$, (in that case $Q = 1,000$) chosen randomly from $\mathcal{U}_1$ was created, and for each of these scenarios the recovery costs were calculated by solving

$$\min \sum_{i\in\mathcal{E}} \pi_i^q - \pi_i$$

$$\text{s.t.} \quad \pi_j^q - \pi_i^q \geq l_{ij}^q \quad \forall\, (i,j) \in \mathcal{A}$$
$$\pi_i^q \geq \pi_i \quad \forall\, i \in \mathcal{E}..$$

Afterwards, the average of these objective values was taken.

- Worst-case costs when recovering to optimality: As for the calculation of the recovery costs, scenarios l^q for $q = 1, \ldots, Q$ were created. Then the respective timetable problem TT(l^q) was solved and the d_1-distance to the given solution measured. The maximum of these distances is the optimality distance, an approximation to the d_1 radius.

- Feasibility: A large number of scenarios is chosen at random by an exponential distribution of average 0.1. We did not choose uniform distribution, as solutions easily tend to be infeasible and less insight is gained. For every scenario we tested if the robust solution is feasible or not, and averaged the feasibility.

- Running times.

Evaluation.

Objective value. In Figure 4.3 the objective values of the robustness concepts for $\mathcal{U}_1$ and $\mathcal{U}_2$ are plotted against the control parameter s, describing the increasing uncertainty of the input data. As expected, the values of the nominal solution are constant throughout s, just like the buffered and the lightly robust solution. The fastest growing costs are those of the strictly robust solution. They might be still acceptable for the small disturbances of $\mathcal{U}_1$, but they are clearly far too high for $\mathcal{U}_2$. The costs of the recovery robust solutions are moderate in both cases. Concerning the (RecOpt-TT) solutions, the costs grow moderately, though a bit faster than those of the recovery robust solution, on $\mathcal{U}_1$, while they stay extremely low for $\mathcal{U}_2$.

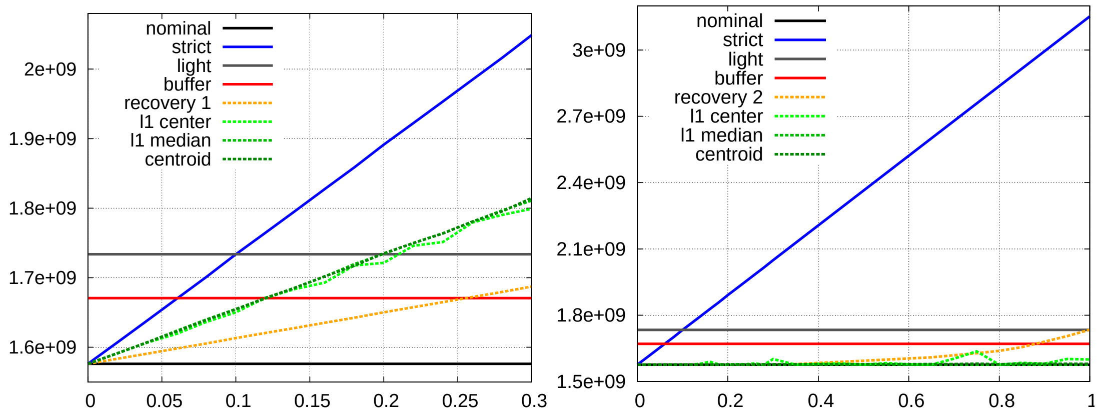


Figure 4.3 Objective function for $\mathcal{U}_1$ (left) and $\mathcal{U}_2$ (right) solutions against s.

Average buffer. The average buffers are shown in Figure 4.4. Most strikingly, the recovery robust solution for $\mathcal{U}_1$ has even larger buffers than the strictly robust solution, which is due to the fact that less weighted edges are buffered more. The lightly robust solution shows an interesting behavior by being not monotone. The centroid, d_1 center and median show a much larger increase in buffer times for $\mathcal{U}_1$ than for $\mathcal{U}_2$.

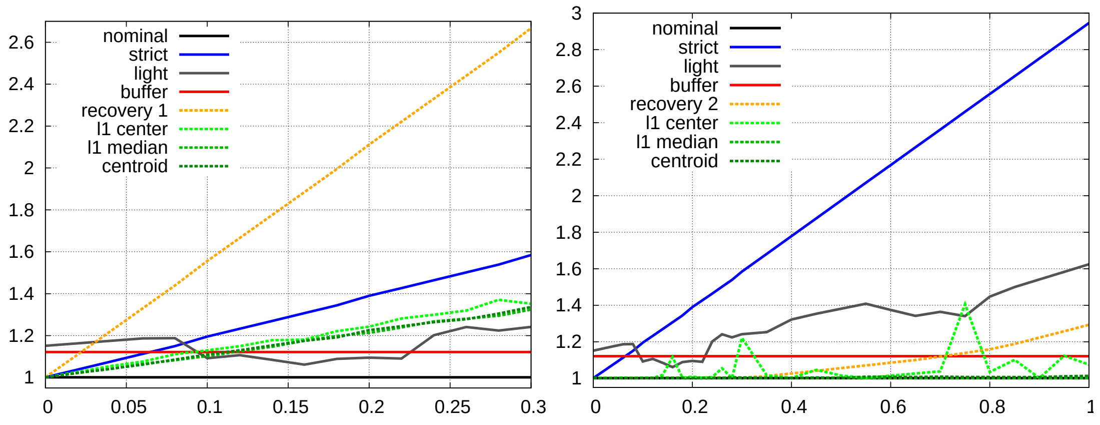


Figure 4.4 Average buffer for $\mathcal{U}_1$ (left) and $\mathcal{U}_2$ (right) against s.

Average recovery costs when recovering to feasibility. The recovery costs for $\mathcal{U}_1$ and $\mathcal{U}_2$ algorithms are depicted in Figure 4.5. Note the larger scale of the right figure: Recovery costs are generally much higher for $\mathcal{U}_2$-type uncertainties. The nominal solution performs worst for $\mathcal{U}_1$, being followed by the buffered solution with a constant offset stemming from the added 6 percent to activity durations. The recovery costs of the lightly robust solution stay a little below those of the recovery robust solution, while the (RecOpt-TT) solutions show the slightest increase. On the other hand, they perform similar to the nominal solution for $\mathcal{U}_2$. Here the recovery robust solution has slightly lower costs, being exceeded by the buffered and especially the light solutions still.

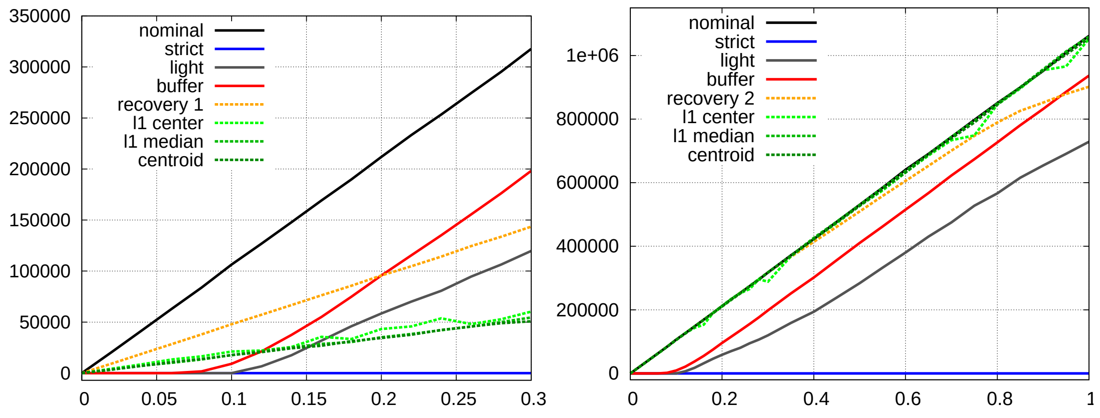


Figure 4.5 Average recovery costs to feasibility for $\mathcal{U}_1$ (left) and $\mathcal{U}_2$ (right) against s.

Worst-case recovery costs when recovering to optimality. Figure 4.6 shows the approximate maximum d_1-distances to the optimal solutions of the uncertainty set. The (RecOpt-TT) solutions perform very good in this category which shows that our heuristic approach (RecOpt-TT) can be used to minimize this distance. For $\mathcal{U}_1$ the solutions gained by (RecOpt-TT) clearly outperform the other robust solutions while they are comparable with some others for $\mathcal{U}_2$. Note that the strict robust solution performs poorly under this measure, as solutions are generally over-buffered.

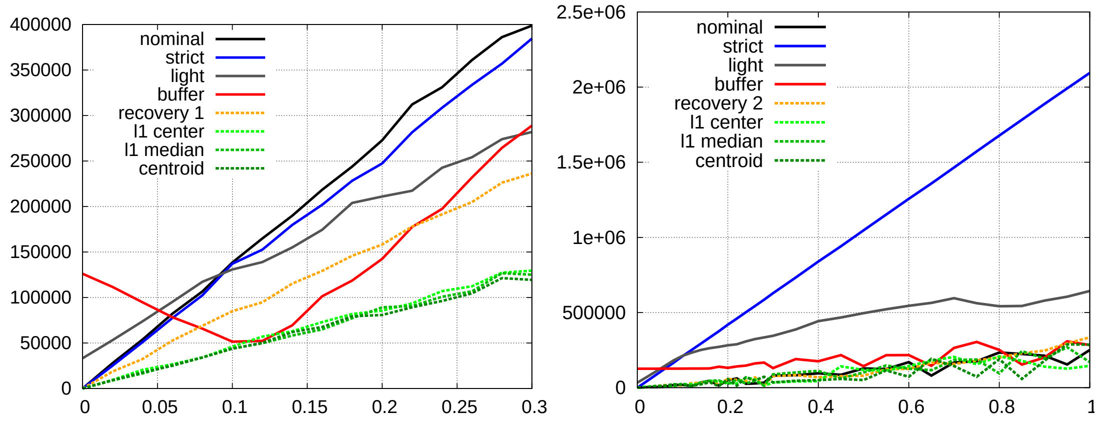


Figure 4.6 Worst-case recovery costs to optimality for $\mathcal{U}_1$ (left) and $\mathcal{U}_2$ (right) against s.

Feasibility. Figure 4.7 shows the average feasibility under exponential scenario distribution. Note that all solutions except of the strictly robust solution strongly decrease their feasibility for growing s in $\mathcal{U}_1$. The lightly robust solution becomes infeasible as soon as its budget is completely used, which is exactly when its objective value equals the strictly robust solution (see Fig. 4.3). Only the centroid keeps a small probability of feasibility throughout all values of s. For the $\mathcal{U}_2$ uncertainty, the situation changes completely. The buffered and the lightly robust solution keep moderate feasibility even for high values of s, while all other solutions (except of the strictly robust) stay low. This is exactly the intension of the recovery-robust approaches: They improve their nominal quality by allowing a repair phase and hence not aiming at feasibility for all scenarios.

Running times. Figure 4.8 shows the running times of the algorithms. Most time-consuming were the calculations of the d_1 center followed by the d_1 median and (R2-TT). The higher running times for the d_1-median and the centroid are due to the presolving phase in which the optimal solutions of all scenarios in S needs to be calculated.

4.2.5 Conclusion

We applied the most prominent robustness to timetabling and compared them on a real-world instance. Furthermore, we introduced a new approach, minimizing the recovery distances to a subset of scenarios, that is easily applicable to any robustness

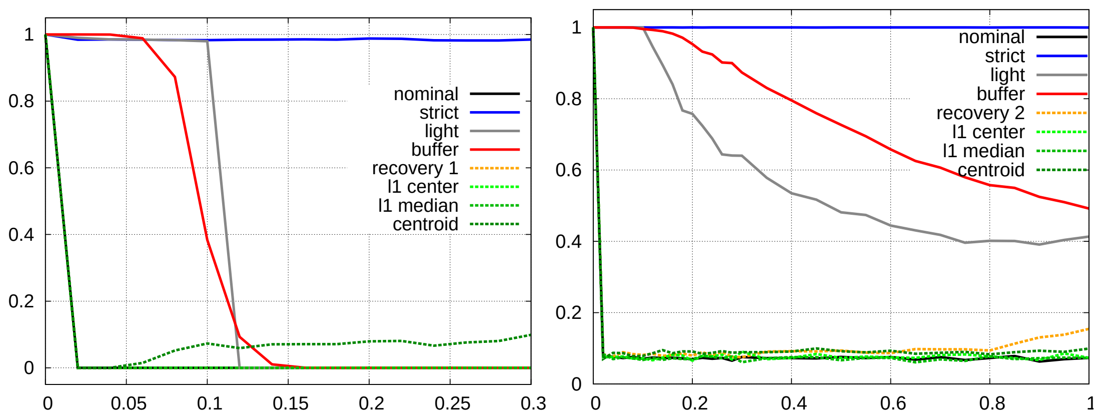


Figure 4.7 Feasibility exponentially distributed for $\mathcal{U}_1$ (left) and $\mathcal{U}_2$ (right) against s, $\mu = 0.1$.

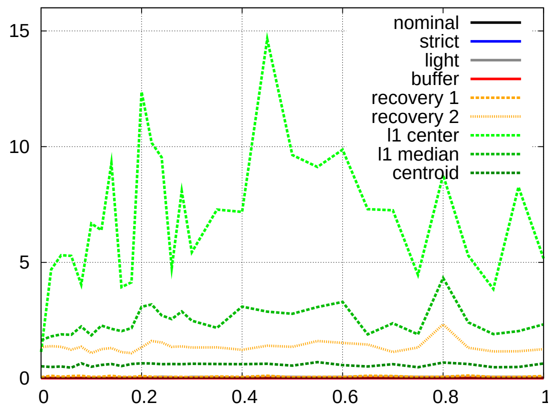


Figure 4.8 Running times in seconds against s.

problem, whenever a method for solving the original problem is at hand. We have shown that there are significant differences in the performance of the concepts depending on the type of uncertainty under consideration. Strict robustness, as an example, is a considerable concept for $\mathcal{U}_1$ uncertainty, but not an option for $\mathcal{U}_2$. Concerning the (RecOpt-TT) solutions, especially the centroid approach gives good feasibility and recovery properties with average costs on $\mathcal{U}_1$, while the same approach for U_2 sticks too closely to the nominal solution for having good robustness properties. We conclude that it is crucial to choose the robustness concept to be applied to the specific problem structure and the uncertainty set.

5 Discrete Problem Applications

Until now, we have focused on continuous problem instances, in which we were able to add a large computational overhead for robustness purposes. However, this is not possible for many discrete problems, for which already the nominal case is computationally challenging. Therefore, we carefully adapt our robustness approaches to the resources available. We consider the problem of loadplanning in intermodal transport in Section 5.1, Steiner trees in Section 5.2, periodic timetables in Section 5.3, and timetable information in Section 5.4.

5.1 Robust Load Planning in Intermodal Transport

5.1.1 Introduction

Container-based transportation is a growing market. The transportation volume of maritime container transport as well as the continental intermodal transport has increased significantly in recent years. Intermodal transport uses different transportation modes like roads, rails or ships. The goods are transported in load units (containers, swap bodies or trailers) and change their transportation modes at bi- or trimodal terminals. Most of the research has been carried out in the field of maritime container terminals [SV08], whereas fewer research concerns road-rail or hinterland terminals. The huge transportation volume leads also to an emerging research field of intermodal rail-truck freight transport within the area of operations research [BMT04, BFJP12].

Here we study how different approaches of robustness can be applied to the load planning problem of trains. For this purpose we use a model of load planning developed in Bruns and Knust [BK12]. The objective of the load planning problem is to assign load units to the wagons of a train and to choose compatible configurations while weight restrictions and other restrictions have to be respected. A configuration defines how many and which types of load units can be placed on a wagon. Weight restrictions limit the payload of the wagons and the whole train. The aim of load planning is to maximize the utilization of the train and to minimize the total costs in the terminal. Contributions to the total costs are given by transportation costs for the load units (for the transport from their current storage location to the assigned wagon) and setup costs for changing the configurations of wagons.

The quality of the input data of the load planning problem differs depending on the time of planning. We consider two different planning situations: pre-loading planning and planning-while-loading. The pre-loading planning phase takes place before train loading begins. The information on the wagons of the train and the load units that should be placed onto the train are given to the terminal operator by booking systems of the railway companies. In this phase, the configurations of the wagons are fixed and a preliminary assignment from load units to wagons is determined. For the planning-while-loading process we assume that all information on load units and the train are verified and possible variations to the information provided by the booking systems are corrected. In this phase, there is no uncertainty left and the load units are finally assigned to the wagons.

The aim of this section is to analyze different robustness concepts for the load planning problem. We focus on the concept of *strict robustness* and compare it with the less conservative approach of *adjustable robustness.* For both concepts and for different types of uncertainties we develop the respective robust counterparts based on a mixed-integer linear programming (MIP) formulation. We show how these formulations can be solved within a reasonable runtime and discuss the resulting robust solutions and their usefulness for our application.

The remainder is organized as follows: In Section 5.1.2 we introduce the considered problem, give a MIP formulation and specify different uncertainties. While in Section 5.1.3 we discuss the concept of strict robustness, in Section 5.1.4 we deal with adjustable robustness. After presenting computational experiments in Section 5.1.5 we conclude with some remarks in Section 5.1.6.

5.1.2 Problem Model

In this section we describe the load planning problem for trains in a container terminal in more detail. We use a slightly modified version of the third mixed-integer linear programming formulation of Bruns and Knust [BK12], which can be easily solved by current MIP solvers.

In Section 5.1.2.1, we start by assuming that all information needed are given exactly and formulate a mixed-integer programming model for the load planning problem. In Section 5.1.2.2 we introduce a reformulated version. Afterwards, we extend the problem to an uncertain optimization problem specifying which parameters are usually uncertain and which values they can take – see Section 5.1.2.3.

5.1.2.1 A MIP-Formulation for Load Planning

In this section we assume that all information needed to model the load planning problem are known. The load units are divided into three different categories: containers, swap bodies, and trailers. In the following, swap bodies are treated as containers because they differ from containers only in the possibility of stacking, which is not relevant here. Containers are 3-dimensional boxes that have four so-called corner castings which are one part of the connector between containers and trucks, trains or ships. The other part of the connection in rail transport are flexible pins on the wagons. The distances in between the corner castings are standardized to 20 feet (5853 $\pm$ 3 mm), 30 feet (8918 $\pm$

4 mm) and 40 feet (11985 ± 5 mm). Note that the corner castings do not have to be at the corners of a container, some containers – and many swap bodies – have an overhang above the corner castings. We categorize load units in four types $t \in \mathcal{T} := \{1, \dots, 4\}$: containers with 20, 30 and 40 ft corner castings distance and trailers.

There are n (approximately 80 in practice) load units $i = 1, \dots, n$ with lengths ℓ_i, weights g_i and overhangs over the corner castings u_i; we assume that the overhangs to both sides are similar. Let $\mathcal{N} := \{1, \dots, n\}$ be the set of all load units, all load units of the same type $t \in \mathcal{T}$ are grouped into sets $\mathcal{N}_t$.

Given is also an empty train consisting of m (approximately 20 to 30) wagons $j \in \mathcal{M} := \{1, \dots, m\}$ where each wagon j belongs to a wagon type $c(j) \in \mathcal{W}$. The train has a given weight limit G which may not be exceeded by the total weight of the assigned load units. The wagons can be used in different configurations which determine how many and which load units can be loaded onto slots of a wagon. $\mathcal{S}_\tau$ denotes the set of all possible slots on a wagon of type τ. Let $\mathcal{K}_\tau$ be the set of all valid physical configurations for a wagon of type $\tau \in \mathcal{W}$. For a wagon j in configuration $k \in \mathcal{K}_{c(j)}$ the binary coefficients α_{tks} encode if load unit type $t \in \mathcal{T}$ fits onto slot $s \in \mathcal{S}_\tau$:

$$\alpha_{tks} = \begin{cases} 1, & \text{if in configuration } k \text{ load unit type } t \text{ fits onto slot } s \\ 0, & \text{otherwise.} \end{cases}$$

For a wagon j in configuration $k \in \mathcal{K}_{c(j)}$ the coefficients $\beta_{ks} \in \mathbb{R}$ denote the maximum allowed overhang of a load unit on slot $s \in \mathcal{S}_\tau$. For each wagon an initial configuration is given. If a wagon setting is changed to another configuration, setup costs occur. The binary coefficients $\kappa^0_{jk} \in \{0, 1\}$ for wagons $j \in \mathcal{M}$ and all configurations $k \in \mathcal{K}_{c(j)}$ for the corresponding wagon type $c(j)$ indicate whether wagon j is initially used in configuration k or not.

We assume that all load units are placed in a storage area of the terminal. For each $i \in \mathcal{N}$ and each $j \in \mathcal{M}$ we denote by d_{ij} the transportation costs for loading load unit i onto wagon j, mainly influenced by the distance between the storage place and the wagon.

Beside the length restrictions we have to consider the following three weight conditions:

(W1) The payload per bogie is restricted according to the wagon bogies and the track the train runs on.

(W2) The payload on a bogie may not be larger than three times the payload on the other bogie – otherwise, the probability for wagons jumping the rails would increase.

(W3) The payload for each slot on the wagon is limited due to stability reasons.

To model the wagon weight restrictions we calculate the wagon bogie loads by the lever principle. Figure 5.1 shows a wagon with two bogies A and B ,and two load units i_1 and i_2. The centers of mass of the load units are assumed to be in the middle of the

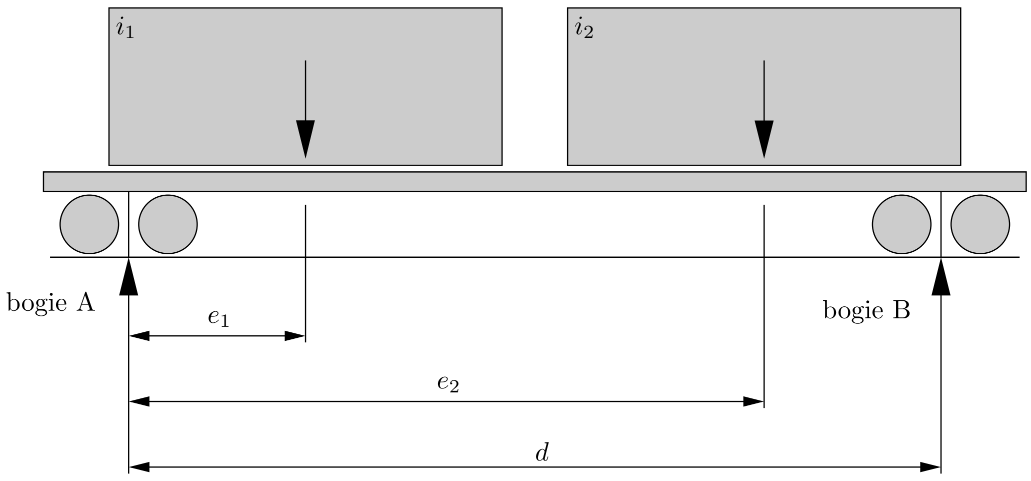


Figure 5.1 Lever principle for weight distributions of wagon bogies.

load units, symbolized by arrows. In the figure three lengths are shown: The distance d in between the bogie attachments, and the levers e_1 and e_2 of the two load units in relation to bogie A. Let g_1 and g_2 be the weights of the two load units and W be the weight of the empty wagon. For this situation, the payload a of bogie A can be calculated as

$$a = \frac{d - e_1}{d} \cdot g_1 + \frac{d - e_2}{d} \cdot g_2 + \frac{W}{2}, \tag{5.1}$$

symmetrically, for bogie B the resulting payload b is given by

$$b = \frac{e_1}{d} \cdot g_1 + \frac{e_2}{d} \cdot g_2 + \frac{W}{2}. \tag{5.2}$$

If more than two load units are loaded onto a wagon, for each additional load unit i an additive term $\frac{d-e_i}{d} \cdot g_i$ for bogie A and $\frac{e_i}{d} \cdot g_i$ for bogie B has to be taken into account.

For the MIP-formulation we need the following parameters to model the weight restrictions. For a wagon of type τ the coefficient $\gamma_\tau \in \mathbb{R}$ denotes the maximum payload for its bogies, the parameter $d_\tau \in \mathbb{R}$ denotes the distance in between the bogie attachments, and $t_\tau \in \mathbb{R}$ denotes the tare mass of the wagon. Furthermore, for a wagon j of type τ and a slot $s \in \mathcal{S}_\tau$ the coefficient $\delta_{\tau s} \in \mathbb{R}$ denotes the maximum payload for slot s and the value $e_{\tau s} \in \mathbb{R}$ is the lever for a load unit on slot s relating to the first bogie.

We introduce the following decision variables in order to choose configurations for all wagons and to assign load units to slots on the wagons:

- $y_{jk} \in \{0, 1\}$ for $j \in \mathcal{M}, k \in \mathcal{K}_{c(j)}$ with

$$y_{jk} = \begin{cases} 1, & \text{if configuration } k \text{ is chosen for wagon } j \\ 0, & \text{otherwise.} \end{cases}$$

- $x_{ijs} \in \{0,1\}$ for $i \in \mathcal{N}, j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}$ with

$$x_{ijs} = \begin{cases} 1, & \text{if load unit } i \text{ is assigned to slot } s \text{ on wagon } j \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, we introduce auxiliary variables $a_j, b_j \in \mathbb{R}$ measuring the bogie payloads of wagon j. While a_j denotes the payload for the front bogie of wagon j, b_j is the payload of the rear bogie. Then the mixed-integer linear program reads:

$$\textbf{(LP)} \quad \max \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_{c(j)}} (w_1 + w_2 \cdot \ell_i + w_3 \cdot g_i) \cdot x_{ijs} \tag{5.3}$$

$$- w_4 \, (m - \sum_{j\in\mathcal{M}} \sum_{k\in\mathcal{K}_{c(j)}} \kappa^0_{jk} y_{jk}) \tag{5.4}$$

$$- w_5 \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_{c(j)}} d_{ij} \cdot x_{ijs} \tag{5.5}$$

s.t.

$$\sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_{c(j)}} x_{ijs} \leq 1 \qquad (i \in \mathcal{N}) \tag{5.6}$$

$$\sum_{k\in\mathcal{K}_{c(j)}} y_{jk} = 1 \qquad (j \in \mathcal{M}) \tag{5.7}$$

$$\sum_{i\in\mathcal{N}_t} x_{ijs} - \sum_{k\in\mathcal{K}_{c(j)}} \alpha_{tks} \cdot y_{jk} \leq 0 \qquad (j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}, t \in \mathcal{T}) \tag{5.8}$$

$$\sum_{i\in\mathcal{N}} u_i \cdot x_{ijs} - \sum_{k\in\mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \leq 0 \qquad (j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}) \tag{5.9}$$

$$a_j - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} = \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.10}$$

$$b_j - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} = \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.11}$$

$$a_j \leq \gamma_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.12}$$

$$b_j \leq \gamma_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.13}$$

$$a_j - 3 \cdot b_j \leq 0 \qquad (j \in \mathcal{M}) \tag{5.14}$$

$$b_j - 3 \cdot a_j \leq 0 \qquad (j \in \mathcal{M}) \tag{5.15}$$

$$\sum_{i\in\mathcal{N}} g_i \cdot x_{ijs} \leq \delta_{c(j),s} \qquad (j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}) \tag{5.16}$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot x_{ijs} \leq G \tag{5.17}$$

$$x_{ijs} \in \{0,1\} \qquad (i \in \mathcal{N}, j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}) \tag{5.18}$$
$$y_{jk} \in \{0,1\} \qquad (j \in \mathcal{M}, k \in \mathcal{K}_{c(j)}) \tag{5.19}$$
$$a_j, b_j \in \mathbb{R} \qquad (j \in \mathcal{M}) \tag{5.20}$$

The parameters $w_1, \dots, w_5 \geq 0$ in the objective function are coefficients weighting the different components. While w_1, w_2, w_3 weight the total number, total length and total weight of assigned load units in (5.3), w_4 is the factor for the setup costs in (5.4) and w_5 is the factor for the transportation costs in (5.5). The setup costs for changing a configuration are assumed to be constant, i.e. in (5.4) the number of changed configurations is minimized. We will write $obj(x, y)$ for the objective function, when the coefficients are clear from the context.

Constraints (5.6) ensure that each load unit is assigned to at most one slot, (5.7) guarantees that for each wagon one associated configuration is chosen. Constraint (5.8) ensures that the types of all assigned load units are feasible, while constraints (5.9) represent the limits of the overhangs. Due to (5.10) and (5.11) the auxiliary variables a_j, b_j are set to the correct values. The payload of all bogies is restricted due to (5.12) and (5.13) (W1). Due to (5.14) and (5.15) the payload on a bogie may not be larger than three times the payload on the other bogie (W2). Constraint (5.16) limits the maximum slot payloads (W3). Finally, the total weight limit of the train is modeled by (5.17). The domains of the variables are defined in (5.18) to (5.20).

Example 5.1. *We consider a small instance with $n = 6$ load units belonging to 3 different types, $m = 3$ wagons belonging to 2 wagon types, and 2 slots for each wagon. We assume that the load units have the types $(1, 1, 1, 2, 2, 3)$, the length of the overhangs are $(100, 100, 200, 100, 200)$, and the weights (in tons) are $(10, 24, 26, 36, 38, 35)$. The first and second wagon are of type one, the third is of type two. We assume that the distances in between the bogie attachments are $d_1 = d_2 = 11200$ for all wagons. Furthermore, a wagon of type one has an empty weight of $W_1 = 16$ and a wagon of type two has a weight of $W_2 = 17$ tons.*

Table 5.1 shows information on the possible configurations where every row corresponds to a configuration $k \in \{1, \dots, 5\}$. Each row contains the number of the configuration k and the corresponding wagon type. Furthermore, for the two possible slots s on each wagon the feasible load unit types, the maximum allowed overhangs β_{ks}, the levers e_{ks}, and the maximum payloads (in tons) are given.

A feasible solution where the load units 1,2,4, and 5 are loaded, is presented in Table 5.2. For the three wagons the chosen configurations as well as the assigned load units are listed. The solution respects all length constraints for the configurations: for each wagon a feasible configuration is chosen (constraint (5.7)), the types of the assigned load units are compatible with it (constraint (5.8)), and the overhangs of all load units are feasible (constraint (5.9)).

Finally, we consider the weight restrictions (W1)-(W3) (constraints (5.12) to (5.16)) and show exemplary that the load of the first wagon is feasible with respect to these constraints. We have to consider the values $d = 11200$, $e_1 = 1500$, $e_2 = 8535$, $W = 16$

configu-ration	wagon type	slot 1				slot 2			
		type	β_{k1}	e_{k1}	payload	type	β_{k2}	e_{k2}	payload
1	1	{1}	200	1500	23	{1}	200	8535	24
2	1	{2}	150	5600	36	{}			
3	1	{3}	200	5600	38	{}			
4	2	{1}	200	1600	18	{1}	200	8500	26
5	2	{2}	100	5600	38	{}			

Table 5.1 Possible configurations of the wagons.

wagon	configuration	load units	
		slot 1	slot 2
1 (type 1)	1	1 (type 1)	2 (type 1)
2 (type 1)	2	4 (type 2)	
3 (type 2)	5	5 (type 2)	

Table 5.2 A feasible solution.

(recall Figure 5.1 for the meaning of these parameters). The allowed payload per bogie is assumed to be $\gamma = 40$, and the maximum payloads for the two slots are $\delta_1 = 23$ and $\delta_2 = 24$. The bogie loads of the first wagon can be calculated as $a = \frac{d-e_1}{d} \cdot g_1 + \frac{d-e_2}{d} \cdot g_2 + \frac{W}{2} = 22.37$, and $b = \frac{e_1}{d} \cdot g_1 + \frac{e_2}{d} \cdot g_2 + \frac{W}{2} = 27.63$. The solution is feasible because

(W1) the bogie loads a, b are not greater than $\gamma = 40$,

(W2) $1/3 \cdot a = 7.46 \leq b = 27.63 \leq 3 \cdot a = 67.11$,

(W3) the weights $g_1 = 10, g_2 = 24$ of the assigned load units do not exceed the maximum allowed weights $\delta_1 = 23$ and $\delta_2 = 24$ for the slots.

5.1.2.2 Problem Reformulation

The presented problem formulation has two drawbacks from the point of view of robust optimization: Firstly, equality constraints as given by (5.10) and (5.11) circumvent the possibility to find a solution that is feasible for every possible scenario. Secondly, it should be possible to determine the values of auxiliary variables like a_j and b_j when scenario information becomes known, as they do not represent a choice of the decision maker, but are introduced only to contain information within the MIP. Therefore, we give an equivalent reformulation in which equality constraints and auxiliary variables are removed.

By combining equations (5.10) and (5.11) with the inequalities (5.14) and (5.15) (which eliminates the auxiliary variables a_j and b_j), for all $j \in \mathcal{M}$ we get

$$\left(\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \frac{t_{c(j)}}{2}\right) - 3\left(\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \frac{t_{c(j)}}{2}\right) \leq 0$$
$$\left(\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \frac{t_{c(j)}}{2}\right) - 3\left(\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \frac{t_{c(j)}}{2}\right) \leq 0$$

Reformulation yields

$$\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \leq t_{c(j)} \qquad (j \in \mathcal{M}) \qquad (5.21)$$
$$\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{4e_{c(j),s} - 3d_{c(j)}}{d_{c(j)}} \cdot x_{ijs} \leq t_{c(j)} \qquad (j \in \mathcal{M}) \qquad (5.22)$$

and, for inequalities (5.12)-(5.13):

$$\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \leq \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \qquad (5.23)$$
$$\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}_{c(j)}} g_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \leq \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \qquad (5.24)$$

We hence obtain:

Lemma 5.2. *Constraints (5.10)-(5.15) are equivalent to constraints (5.21)-(5.24).*

Finally, to handle the problem constraints easier, we introduce a more compact problem formulation. We split the constraints into three classes: Those containing only x-variables, constraints containing only y-variables, and constraints containing both. We obtain

$$\begin{aligned} Ay \quad\quad\;\; &= p && (5.25)\\ Bx &\leq q && (5.26)\\ Cy + Dx &\leq r && (5.27)\\ x, y &\text{ binary} && (5.28) \end{aligned}$$

where (5.25) represents constraint (5.7), while (5.26) represents constraints (5.6), (5.16), (5.17), (5.21)-(5.24). The coupling inequality (5.27) covers constraints (5.8) and (5.9).

5.1.2.3 Specifying the Uncertainty

As described in Section 5.1.1, almost inevitably the parameters used during problem planning will differ from those that come up during operation. This calls for planning in a robust way. In order to set up a robust optimization approach for load planning we first have to identify which parameters are uncertain and which values they may take. This is done in the following by specifying *uncertainty sets*. Note that in practice nearly all parameters may be uncertain, but the following types are most common:

1. The lengths of the overhangs of load units sometimes differ from the ones announced by the booking system. Hence, we assume that the parameter u_i for each load unit i varies in an interval $[u_i^{min}, u_i^{max}]$, which results in the following uncertainty set:
$$\mathcal{U}_1 = \{u \in \mathbb{R}^n : u^{min} \leq u \leq u^{max}\}.$$

2. Often the weights g_i of the load units differ, which may be modeled by the uncertainty set
$$\mathcal{U}_2 = \{g \in \mathbb{R}^n : g^{min} \leq g \leq g^{max}\}.$$
Since it is very unlikely that the weights of all load units vary, analogously to the approach of Bertsimas and Sim [BS04] we also consider the situation where only a limited number $\Gamma \in \{0, \dots, n\}$ of load unit weights may differ from the expected weights. If we denote by $\tilde{g}$ the expected (*nominal*) weights, the uncertainty set
$$\begin{aligned}\mathcal{U}_2^\Gamma = \{&g \in \mathbb{R}^n : \exists I \subseteq \mathcal{N}, |I| \leq \Gamma,\\ &g_i = \tilde{g}_i \text{ for } i \notin I,\ g_i^{min} \leq g_i \leq g_i^{max} \text{ for } i \in I\}\end{aligned}$$
says that the weights of at most Γ load units differ in their given intervals. Note that for $\Gamma = n$, the set $\mathcal{U}_2^\Gamma$ equals $\mathcal{U}_2$.
For the computational results we analyze two variants of $\mathcal{U}_2^\Gamma$. Normally we assume that the weights g_i of the load units are distributed around the nominal values. In practice, sometimes also the extreme case occurs that a load unit which originally has been booked as empty, is loaded. If $\overline{g}_i$ denotes the maximum feasible payload of load unit i, we model this situation as a special case of $\mathcal{U}_2^\Gamma$ with $g_i^{min} := \tilde{g}_i$ (a load unit does not become lighter) and $g_i^{max} := \overline{g}_i$:
$$\begin{aligned}\overline{\mathcal{U}}_2^\Gamma := \{&g \in \mathbb{R}^n : \exists I \subseteq \mathcal{N}, |I| \leq \Gamma,\\ &g_i = \tilde{g}_i \text{ for } i \notin I,\ \tilde{g}_i \leq g_i \leq \overline{g}_i \text{ for } i \in I\}.\end{aligned}$$
Note that $\overline{g}_i$ does not depend on the nominal weight $\tilde{g}_i$ since it is mainly influenced by the size of the load unit. In principal such deviations might happen to any load unit.

3. Sometimes a wagon has some kind of bug, and as a consequence cannot be assigned any load unit. This uncertainty is not reflected in the problem's parameters used in (5.6)-(5.20). We hence introduce a *bug parameter* $f_j \in \{0, 1\}$ which

indicates that wagon j has a bug when f_j is zero and change constraint (5.8) to

$$\sum_{i\in\mathcal{N}_t} x_{ijs} - \sum_{k\in\mathcal{K}_{c(j)}} \alpha_{tks}\cdot f_j \cdot y_{jk} \leq 0 \qquad (j\in\mathcal{M}, s\in\mathcal{S}_{c(j)}, t\in\mathcal{T}) \tag{5.29}$$

This ensures that no load units are assigned to wagon j if it has a bug. Similar to the uncertainty $\mathcal{U}_2$, it might be unrealistic to assume that all wagons have a bug at the same time. Hence, we define the uncertainty set as

$$\mathcal{U}_3^\Gamma = \{f\in\{0,1\}^m : \sum_{j\in\mathcal{M}} f_j \geq m-\Gamma\}$$

for a parameter $\Gamma\in\{0,\dots,m\}$ saying that at most Γ wagons may have a bug.

Note that for uncertainties $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$ the case $\Gamma = 0$ corresponds to the situation where no parameter is uncertain. These cases are equivalent to the nominal problem where robustness is not considered. We do not study an uncertainty set $\mathcal{U}_1^\Gamma$ because the uncertainty in parameter u only affects constraints concerning individual load unit assignments, but no groups of load unit assignment. So for $\Gamma > 0$ we have $\mathcal{U}_1^\Gamma = \mathcal{U}_1$. For the uncertainty $\mathcal{U}_3^\Gamma$ the case $\Gamma = m$ corresponds to the situation that all wagons may have a bug; according to our previous definitions this could also be notated as $\mathcal{U}_3$. This set is not meaningful, since it contains the worst-case scenario $f_j = 0$ for all $j\in\mathcal{M}$, i.e., all wagons have a bug, which does not allow any assignment of load units. In the following robustness models, we only consider the nominal objective function; i.e., we may assume that all uncertainties above only affect the *constraints* of the load planning problem, not the *objective*. Note that considering a worst-case maximization problem instead would only make a difference for $\mathcal{U}_2$ and $\mathcal{U}_2^\Gamma$, as g is the only uncertain parameter of the objective function.

To illustrate the uncertainty sets we apply some of them to the example introduced at the end of Subsection 5.1.2.1. Considering uncertainty set $\mathcal{U}_1$ if the overhangs of load unit one and two are of the interval $[100, 200]$, the solution described in Table 5.2 is feasible for all scenarios. If we consider $\mathcal{U}_2$ and weights from the interval $[23, 25]$, the solution of Table 5.2 is no longer feasible for all scenarios.

Having specified different types of uncertainties for the load planning problem, we will in the following write $LP(\xi), \xi\in\mathcal{U}$ to indicate that the problem depends on the unknown parameters $\xi\in\mathcal{U}$. As an example, $LP(g), g\in\mathcal{U}_2$ indicates that the weights g are uncertain and stem from the uncertainty set $\mathcal{U}_2$. We furthermore specify one nominal scenario $\tilde{\xi}\in\mathcal{U}$ for each of our uncertainty sets: The nominal scenario $\tilde{u}\in\mathcal{U}_1$ contains the overhangs announced in the booking system, and the nominal scenario for $\mathcal{U}_2$ and $\mathcal{U}_2^\Gamma$ is given by the nominal weight values $\tilde{g}_i$. The nominal scenario for $\mathcal{U}_3^\Gamma$ assumes that no wagon has a bug, i.e. $\tilde{f}_j = 1$ for all $j\in\mathcal{M}$.

The uncertainty $\mathcal{U}_1$ influences the coefficients of the matrix D in (5.27). We denote this by writing $D(u), u\in\mathcal{U}_1$, to make clear that D is actually a function that maps scenarios to coefficients. Analogously, we write $B(g)$ for $g\in\mathcal{U}_2$, and $C(f)$ for $f\in\mathcal{U}_3^\Gamma$.

5.1.3 Strictly Robust Load Planning

Recall that a solution to an uncertain problem is called *strictly robust* if it is feasible for any of the scenarios in the uncertainty set. Thus, in strictly robust load planning, we would like to hedge against all possible scenarios by calculating a solution (x, y) that is feasible for $LP(\xi)$ for all $\xi \in \mathcal{U}$. The resulting robust problems depend on the type of uncertainty we consider. We denote the strictly robust problem for uncertainty set $\mathcal{U}_i$ by (SR_i).

A typical application for this concept would be when the uncertainty is assumed to be small, i.e., only minor changes in the problem parameters occur, and there is no time during operations to be spent on changing the setup of the wagons or even communicate changes in the load unit assignment to the human crane operators.

5.1.3.1 Uncertainty $\mathcal{U}_1$

If we apply the first type of uncertainty to the problem, only the matrix D in constraints (5.27) is affected. Hence, we have to find (x, y) maximizing $obj(x, y)$ such that

$$Ay \qquad = p \tag{5.30}$$

$$Bx \leq q \tag{5.31}$$

$$Cy + D(u)x \leq r \quad \forall u \in \mathcal{U}_1 \tag{5.32}$$

$$x, y \text{ binary.} \tag{5.33}$$

As all variables x_{ijs} are non-negative, the worst-case for the inequality (5.9) can be directly calculated by using the respective upper bounds of the uncertainty set $\mathcal{U}_1$.

Lemma 5.3. *The strictly robust load planning problem with uncertainty set $\mathcal{U}_1$ is equivalent to the following MIP:*

$$\begin{array}{lll} (SR_1) & \max\, obj(x, y) & \\ & s.t. \displaystyle\sum_{i \in \mathcal{N}} u_i^{max} \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \leq 0 & (j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}) \\ & (5.6) - (5.8), (5.10) - (5.20) & \end{array}$$

Proof. Let $\mathcal{F}$ be the set of feasible solutions for (SR_1), i.e.,

$$\mathcal{F} = \Big\{(x, y) : \sum_{i \in \mathcal{N}} u_i^{max} \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \leq 0 \ \forall j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}, \\ (5.6) - (5.8), (5.10) - (5.20)\Big\},$$

and let

$$\mathcal{F}' = \Big\{(x, y) : \sum_{i \in \mathcal{N}} u_i \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \leq 0 \ \forall j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}, u \in \mathcal{U}_1, \\ (5.6) - (5.8), (5.10) - (5.20)\Big\}$$

be the set of strictly robust solutions w.r.t. $\mathcal{U}_1$. We show $\mathcal{F} = \mathcal{F}'$. Due to $u^{max} \in \mathcal{U}_1$ we have $\mathcal{F} \subseteq \mathcal{F}'$. Now let $(x, y) \in \mathcal{F}'$ and assume there are $\hat{u} \in \mathcal{U}_1$ and j, s such that $\sum_{i \in \mathcal{N}} \hat{u}_i \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} > 0$. As $x \geq 0$, we have $\sum_{i \in \mathcal{N}} \hat{u}_i \cdot x_{ijs} \leq \sum_{i \in \mathcal{N}} u_i^{max} \cdot x_{ijs}$, which is a contradiction. Therefore, $\mathcal{F}' \subseteq \mathcal{F}$ and hence $\mathcal{F} = \mathcal{F}'$. $\square$

Lemma 5.3 shows that (SR_1) is again a load planning problem, where the overhangs are replaced by their respective worst-case values. Therefore, this approach can be solved as efficiently as the nominal problem.

5.1.3.2 Uncertainty $\mathcal{U}_2$

As the uncertainty set $\mathcal{U}_2$ affects constraints (5.26), the problem we consider here consists of finding a solution (x, y) maximizing $obj(x, y)$ such that

$$
\begin{aligned}
Ay \quad\quad\quad &= p && (5.34)\\
B(g)x &\leq q \quad \forall g \in \mathcal{U}_2 && (5.35)\\
Cy + \quad Dx &\leq r && (5.36)\\
x, y &\text{ binary.} && (5.37)
\end{aligned}
$$

For $\mathcal{U}_1$ it was straightforward to see that the worst-case occurs if u_i takes the value u_i^{max}. We now look at $\mathcal{U}_2$: For constraints (5.16) and (5.17) we see that $g_i = g_i^{max}$ is the worst-case. However, this is not so clear when looking at constraints (5.21)-(5.24).

These inequalities can also be interpreted geometrically: If we want to maximize the left-hand side of constraints (5.21), for all wagons j and slots s with $d_{c(j)} - 4e_{c(j),s} \geq 0$ the weights g_i of the assigned load units i have to be chosen as g_i^{max}. These are the load units where the distance from the attachment of bogie A to the center of mass is smaller than $\frac{d_{c(j)}}{4}$, see Figure 5.1. All other load unit weights have to be chosen as g_i^{min}.

Also, if we want to maximize the violation of constraints (5.22), for all wagons j and slots s with $4e_{c(j),s} - 3d_{c(j)} \geq 0$ the weights g_i have to be chosen as g_i^{max}. These are the load units where the distance from the attachment of bogie A to the center of mass is larger than $\frac{3 \cdot d_{c(j)}}{4}$, which is equivalent to the fact that the distance from the attachment of bogie B to the center of mass is smaller than $\frac{d_{c(j)}}{4}$. Again all other load unit weights have to be chosen as g_i^{min}.

Since all variables x_{ijs} and the parameters d_τ are non-negative, for the worst-case scenario we have to distinguish summands where $d_{c(j)} - 4e_{c(j),s}$ is positive and negative, respectively. As described above, if $d_{c(j)} - 4e_{c(j),s}$ is negative, g_i has to be replaced by g_i^{min}; otherwise g_i has to be replaced by g_i^{max}. For the MIP-formulation we define new parameters $\hat{g}^l_{i\tau s} \in \mathbb{R}$ for $i \in \mathcal{N}, \tau \in \mathcal{W}, s \in \mathcal{S}_\tau$ to replace g_i in constraint (5.21) where

$$\hat{g}^l_{i\tau s} = \begin{cases} g_i^{min}, & \text{if } d_\tau - 4e_{\tau,s} < 0 \\ g_i^{max}, & \text{otherwise.} \end{cases}$$

In constraint (5.22) for the worst-case scenario we replace g_i by $\hat{g}^r_{i\tau s} \in \mathbb{R}$ for $i \in \mathcal{N}, \tau \in \mathcal{W}, s \in \mathcal{S}_\tau$ with

$$\hat{g}^r_{i\tau s} = \begin{cases} g_i^{min}, & \text{if } 4e_{\tau,s} - 3d_\tau < 0 \\ g_i^{max}, & \text{otherwise.} \end{cases}$$

As for all wagons we know that $e_{c(j),s}$ as well as $d_{c(j)} - e_{c(j),s}$ are positive, for constraints (5.23) and (5.24) again the maximum weights g_i^{max} are the worst-case.

By using constraints (5.21) and (5.22) with the above described replacement of g_i we can now formulate a computationally tractable version of the strictly robust problem.

Lemma 5.4. *The strictly robust load planning problem with uncertainty set $\mathcal{U}_2$ is equivalent to the following MIP:*

$$(SR_2) \quad \max\ obj(x, y)$$

s.t.

$$\sum_{i\in\mathcal{N}} g_i^{max} \cdot x_{ijs} \le \delta_{c(j),s} \qquad (j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}) \qquad (5.38)$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_{c(j)}} g_i^{max} \cdot x_{ijs} \le G \qquad (5.39)$$

$$\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \hat{g}^l_{i,c(j),s} \cdot \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \le t_{c(j)} \qquad (j \in \mathcal{M}) \qquad (5.40)$$

$$\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \hat{g}^r_{i,c(j),s} \cdot \frac{4e_{c(j),s} - 3d_{c(j)}}{d_{c(j)}} \cdot x_{ijs} \le t_{c(j)} \qquad (j \in \mathcal{M}) \qquad (5.41)$$

$$\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} g_i^{max} \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \le \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \qquad (5.42)$$

$$\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} g_i^{max} \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \le \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \qquad (5.43)$$

$$(5.6) - (5.9), (5.18) - (5.19)$$

Proof. Let $\mathcal{F}$ be the set of feasible solutions for (SR_2), and let $\mathcal{F}'$ be the set of strictly robust solutions w.r.t. $\mathcal{U}_2$. We show $\mathcal{F} = \mathcal{F}'$. To show that $\mathcal{F} \subseteq \mathcal{F}'$ holds, we give a scenario of $\mathcal{U}_2$ for inequalities (5.39) to (5.40). Due to constraints (5.6) for each $i \in \mathcal{N}$ at most one of the values g_i^{min} and g_i^{max} is used for the calculation of constraints (5.40) and (5.41). So the used values for g equal a scenario of $\mathcal{U}_2$. The same holds for inequalities (5.39) to (5.42) where the values g_i^{max} are used for all $i \in \mathcal{N}$.

Conversely, let $(x, y) \in \mathcal{F}'$ and assume there are $\hat{g} \in \mathcal{U}_2$ and j, s such that one of the inequalities (5.39) to (5.42) is violated. As $x \ge 0$, this would imply that $\sum_{i\in\mathcal{N}} \hat{g}_i \cdot x_{ijs} \le \sum_{i\in\mathcal{N}} g_i^{max} \cdot x_{ijs}$, which is a contradiction. For constraints (5.40) and (5.41) we cannot choose a $\hat{g} \in \mathcal{U}_2$ that enlarges the left hand side because the values $\hat{g}^l_{i,c(j),s}$ and $\hat{g}^r_{i,c(j),s}$

are chosen in a way to maximize the left-hand sides (since they include maximum values for positive summands and minimum for negative ones). Therefore, $\mathcal{F}' \subseteq \mathcal{F}$ and hence $\mathcal{F} = \mathcal{F}'$. □

Note that we did not construct a quasi-worst-case scenario in the sense of Section 3.3.2, as the same parameter g_i is chosen differently between the constraints.

5.1.3.3 Uncertainty $\mathcal{U}_2^\Gamma$

We now consider the problem of finding (x, y) maximizing $obj(x, y)$ such that

$$\begin{aligned} Ay \quad\quad\quad &= p &&(5.44)\\ B(g)x &\le q \quad \forall g \in \mathcal{U}_2^\Gamma &&(5.45)\\ Cy + \quad Dx &\le r &&(5.46)\\ x, y &\text{ binary.} &&(5.47)\end{aligned}$$

First, we consider a single uncertain constraint of the type

$$\sum_{j=1}^{n} B_{ij}(g)x_j \le q_i \quad \forall g \in \mathcal{U}_2^\Gamma \tag{5.48}$$

for an arbitrary but fixed row i and assume that we have variables x_j for $j = 1, \ldots, n$. Similar to [BS04], we determine the robust counterpart of this constraint. Let x be fixed. Recall that $\tilde{g}$ is the nominal value of g and let g^{wc} be the worst-case value of g that maximizes the left-hand side of (5.48). Each component g_j^{wc} is either equal to g_j^{min} or g_j^{max} depending on the sign of the corresponding B_{ij}-value. We denote by

$$\beta'(x, \Gamma) = \max \left\{ \sum_{j=1}^{n} (B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j\alpha_j : \sum_{j=1}^{n} \alpha_j \le \Gamma, \ \alpha \in \{0, 1\}^n \right\}$$

the *worst-case* for the left-hand side of constraint (5.48). $\beta'(x, \Gamma)$ delivers the worst possible deviation for a constraint, compared to the nominal parameter when x is fixed. If we relax the integrality constraint for the variable α, we get

$$\beta(x, \Gamma) = \max \left\{ \sum_{j=1}^{n} (B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j\alpha_j : \sum_{j=1}^{n} \alpha_j \le \Gamma, \ 0 \le \alpha \le 1 \right\}. \tag{5.49}$$

It is easy to see that $\beta'(x, \Gamma) = \beta(x, \Gamma)$.

Using formulation (5.49), we conclude that

$$\sum_{j=1}^{n} B_{ij}(\tilde{g})x_j + \beta(x, \Gamma) \le q_i \ \Leftrightarrow \ \sum_{j=1}^{n} B_{ij}(g)x_j \le q_i \text{ for all } g \in \mathcal{U}_2^\Gamma. \tag{5.50}$$

We now consider the dual problem of (5.49):

$$\begin{aligned}\beta^D(x,\Gamma) &= \min\{\Gamma z + \sum_{j=1}^{n} \pi_j : z + \pi_j \geq (B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j \ \forall j = 1,\ldots,n,\\ &\qquad z, \pi_j \geq 0 \ \forall j = 1,\ldots,n\},\end{aligned}$$

where z is the dual variable for constraint $\sum_{j=1}^{n} \alpha_j \leq \Gamma$ and π_j is dual to $\alpha_j \leq 1$. Using strong duality, we conclude:

Lemma 5.5. *Constraint (5.48) for fixed i and a given $x \in \{0,1\}^n$ holds if and only if there are real numbers z, $\pi_1, \ldots, \pi_n$ such that*

$$\begin{aligned}\sum_{j=1}^{n} B_{ij}(\tilde{g})x_j + \Gamma z + \sum_{j=1}^{n} \pi_j &\leq q_i\\ z + \pi_j &\geq (B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j && (j = 1,\ldots,n)\\ \pi_j &\geq 0 && (j = 1,\ldots,n)\\ z &\geq 0\end{aligned}$$

We are now in the position to reformulate the problem:

Theorem 5.6. *For $\Gamma \geq 1$, the strictly robust load planning problem with uncertainty set $\mathcal{U}_2^\Gamma$ is equivalent to the following MIP:*

$$(SR_2^\Gamma) \quad \max\ obj(x,y)$$

s.t.

$$\sum_{i\in\mathcal{N}} g_i^{max} \cdot x_{ijs} \leq \delta_{c(j)s} \qquad (j \in \mathcal{M}, s \in S_{c(j)}) \tag{5.51}$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in S_{c(j)}} \tilde{g}_i \cdot x_{ijs} + \Gamma z^{(5.17)} + \sum_{i\in\mathcal{N}} \pi_i^{(5.17)} \leq G \tag{5.52}$$

$$z^{(5.17)} + \pi_i^{(5.17)} - (g_i^{max} - \tilde{g}_i) \sum_{j\in\mathcal{M}} \sum_{s\in S_{c(j)}} x_{ijs} \geq 0 \qquad (i \in \mathcal{N}) \tag{5.53}$$

$$\sum_{i\in\mathcal{N}} \sum_{s\in S_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \Gamma z_j^{(5.21)} + \sum_{i\in\mathcal{N}} \pi_{ij}^{(5.21)} \leq t_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.54}$$

$$z^{(5.21)} + \pi_{ij}^{(5.21)} - (\hat{g}_{i\tau s}^{l} - \tilde{g}_i) \sum_{s\in S_{c(j)}} \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \geq 0 \qquad (i \in \mathcal{N}, j \in \mathcal{M}) \tag{5.55}$$

$$\sum_{i\in\mathcal{N}} \sum_{s\in S_{c(j)}} \tilde{g}_i \cdot \frac{4e_{c(j),s} - 3d_{c(j)}}{d_{c(j)}} \cdot x_{ijs} + \Gamma z_j^{(5.22)} + \sum_{i\in\mathcal{N}} \pi_{ij}^{(5.22)} \leq t_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.56}$$

$$z^{(5.22)} + \pi_{ij}^{(5.22)} - (\hat{g}_{i\tau s}^{r} - \tilde{g}_i) \sum_{s \in S_{c(j)}} \frac{4e_{c(j),s} - 3d_{c(j)}}{d_{c(j)}} \cdot x_{ijs} \geq 0 \qquad (i \in \mathcal{N}, j \in \mathcal{M}) \tag{5.57}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in S_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \Gamma z_j^{(5.23)} + \sum_{i \in \mathcal{N}} \pi_{ij}^{(5.23)} \leq \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.58}$$

$$z^{(5.23)} + \pi_{ij}^{(5.23)} - (g_i^{max} - \tilde{g}_i) \sum_{s \in S_{c(j)}} \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \geq 0 \qquad (i \in \mathcal{N}, j \in \mathcal{M}) \tag{5.59}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in S_{c(j)}} \tilde{g}_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \Gamma z_j^{(5.24)} + \sum_{i \in \mathcal{N}} \pi_{ij}^{(5.24)} \leq \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.60}$$

$$z^{(5.24)} + \pi_{ij}^{(5.24)} - (g_i^{max} - \tilde{g}_i) \sum_{s \in S_{c(j)}} \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \geq 0 \qquad (i \in \mathcal{N}, j \in \mathcal{M}) \tag{5.61}$$

$$(5.6) - (5.9), (5.18) - (5.19) \tag{5.62}$$

$$z^{(5.17)}, z^{(5.21)}, z^{(5.22)}, z^{(5.23)}, z^{(5.24)} \geq 0 \tag{5.63}$$

$$\pi^{(5.17)}, \pi^{(5.21)}, \pi^{(5.22)}, \pi^{(5.23)}, \pi^{(5.24)} \geq 0 \tag{5.64}$$

Proof. By applying Lemma 5.5 to the constraints (5.17) and (5.21)-(5.24), the formulation follows. Doing so is not necessary for constraints (5.16), since for each j and s at most one of the variables x_{ijs} may be equal to 1. Therefore, in (5.51) we may simply use the worst-case g^{max} for this constraint. □

5.1.3.4 Uncertainty $\mathcal{U}_3^\Gamma$

We now consider the problem of finding a solution (x, y) maximizing $obj(x, y)$ such that

$$Ay = p \tag{5.65}$$

$$Bx \leq q \tag{5.66}$$

$$C(f)y + Dx \leq r \quad \forall f \in \mathcal{U}_3^\Gamma \tag{5.67}$$

$$x, y \text{ binary.} \tag{5.68}$$

As it turns out, this setting is too restrictive to assign any load unit at all.

Lemma 5.7. *The strictly robust load planning problem with respect to the uncertainty set* $\mathcal{U}_3^\Gamma$ *for* $\Gamma \geq 1$ *is equivalent to the MIP*

$$(SR_3^\Gamma) \quad \max\ obj(x, y)$$

$$s.t. \quad \sum_{i \in \mathcal{N}_t} x_{ijs} \leq 0 \qquad (j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}, t \in \mathcal{T}) \tag{5.69}$$

$$(5.6), (5.7), (5.9) - (5.20).$$

Proof. Let $\mathcal{F}$ be the set of feasible solutions of (SR_3^Γ) and let $\mathcal{F}'$ be the set of strictly robust solutions w.r.t. $\mathcal{U}_3^\Gamma$. We show $\mathcal{F} = \mathcal{F}'$. Obviously, we have $\mathcal{F} \subseteq \mathcal{F}'$. On the other hand, let (x, y) be in $\mathcal{F}'$. Assume that for a wagon j and a slot $s \in \mathcal{S}_{c(j)}$ there is a type t and a load unit $i \in \mathcal{N}_t$ such that $x_{ijs} = 1$ holds. Then (x, y) is not feasible for the scenario $f_j = 0$, $f_h = 1$ for $h \neq j$, which is in $\mathcal{U}_3^\Gamma$. Therefore, $x = 0$ for $(x, y) \in \mathcal{F}'$ which implies $\mathcal{F}' \subseteq \mathcal{F}$ and hence $\mathcal{F} = \mathcal{F}'$. □

We have therefore shown that $f = 0$ is a quasi-worst-case scenario.

Lemma 5.8. *In every feasible solution to* (SR_3^Γ) *we must have* $x = 0$*, meaning that no load unit is assigned at all.*

Proof. Let (x, y) be a feasible solution to (SR_3^Γ). From (5.69) we see that for all load units $i \in \mathcal{N}$ we have $x_{ijs} = 0$ for all $j \in \mathcal{M}, s \in \mathcal{S}_{c(j)}$. □

5.1.4 Adjustable Robust Load Planning

In many applications it is unnecessarily restrictive to assume that a solution has to be feasible for every possible scenario from the uncertainty set. Instead, we require only a part of the variables to be fixed beforehand, while we allow the others to be chosen as soon as the scenario becomes known. The former are called “here and now”-variables, the latter “wait and see”, see Section 2.3.

In the context of *adjustable robust load planning*, we search for a solution y, i.e. a configuration of wagons, such that for every scenario ξ from the uncertainty set $\mathcal{U}$ we can find values for x, i.e., the assignment of load units to wagon slots, such that (x, y) is feasible for $LP(\xi)$. Such a solution is called *adjustable robust.* The aim of the adjustable robust counterpart (aSR) is to find an adjustable robust solution that performs best in its worst-case. As before, we refer to the set of adjustable robust solutions with respect to $\mathcal{U}$ as $\mathrm{aSR}(\mathcal{U})$.

For load planning, this approach is motivated by the two planning phases of pre-loading and planning-while-loading as described in Section 5.1.1. As it is time-consuming to change the configuration of a wagon – basically, a worker has to go around the whole train –, but easily accomplished to put a load unit onto another wagon, we assume that the assignment of load units to slots can be done when the realized scenario is revealed.

5.1.4.1 Uncertainty $\mathcal{U}_1$

Assuming that a wagon configuration y is fixed, the set of possible load unit assignments in scenario $u \in \mathcal{U}_1$ is given as

$$\mathcal{F}(y, D(u)) := \Big\{x : Ay = p, Bx \leq q, Cy + D(u)x \leq r\Big\}.$$

The adjustable robust load planning problem with respect to the uncertainty $\mathcal{U}_1$ is then stated as follows:

$$(aSR_1) \quad \max_{y} \min_{u \in \mathcal{U}_1} \max_{x \in \mathcal{F}(y, D(u))} obj(x, y)$$

$$\text{s.t. } \forall u \in \mathcal{U}_1 \; \exists x^{D(u)} \; : \; x^{D(u)} \in \mathcal{F}(y, D(u)).$$

Note that the problem in this formulation is intractable; in fact, there are infinitely many constraints and variables due to the infinite number of scenarios $u \in \mathcal{U}_1$ that need to be considered. Nevertheless, we show that this problem is tractable in a different formulation.

To analyze (aSR_1), we make use of the results on worst-case scenarios as presented in Section 3.3.2. As the objective function is assumed to be certain, every feasibility worst-case scenario is also a global worst-case scenario, and both $Pr_X(\mathbb{SR}) = \mathbb{aSR}$ and $z^{\mathbb{SR}} = z^{\mathbb{aSR}}$ hold.

As has already been demonstrated in the proof of Lemma 5.3, the existence of such a scenario is indeed the case for the considered problem:

Lemma 5.9. *u^{max} is a global worst-case scenario to (aSR_1).*

We can therefore conclude that instead of considering all infinitely many scenarios $u \in \mathcal{U}_1$ for (aSR_1), we only need to consider u^{max}. Therefore, solving (aSR_1) simply amounts to solving (SR_1).

Corollary 5.10. *Let (x^*, y^*) be an optimal solution to (SR_1). Then y^* is an optimal solution to (aSR_1).*

5.1.4.2 Uncertainty $\mathcal{U}_2$

Analogously to the setting in Section 5.1.4.1, the problem we consider here is the following:

$$\begin{aligned} (aSR_2) \quad & \max_{y} \min_{g \in \mathcal{U}_2} \max_{x \in \mathcal{F}(y, B(g))} obj(x, y) \\ & \text{s.t. } \forall g \in \mathcal{U}_2 \; \exists x^{B(g)} \; : \; x^{B(g)} \in \mathcal{F}(y, B(g)), \end{aligned}$$

where

$$\mathcal{F}(y, B(g)) := \Big\{ x : Ay = p, B(g)x \le q, Cy + Dx \le r \Big\}.$$

Unfortunately, as discussed in Section 5.1.3.2, there is no single worst-case scenario as in $\mathcal{U}_1$. From an applied point of view, there is no set of load unit weights that maximally disturbs the balance constraints of all wagon assignments at the same time; from a theoretical point of view, each of the constraints (5.21) and (5.22) has a different worst-case.

Instead of reducing the infinite scenario set $\mathcal{U}_2$ to a single scenario, we therefore consider the following heuristic approach: We generate a finite subset $\mathcal{G} = \{g^1, \ldots, g^N\}$ of scenarios from $\mathcal{U}_2$, and solve

$$(aSR_2)^{\mathcal{G}} \quad \max z$$

$$\begin{aligned}
\text{s.t.}\quad obj(x^i, y) &\geq z && (i = 1, \dots, N)\\
Ay &= p &&\\
B(g^i)x^i &\leq q && (i = 1, \dots, N)\\
Cy + Dx^i &\leq r && (i = 1, \dots, N)\\
x^i, y &\text{ binary} && (i = 1, \dots, N)
\end{aligned}$$

i.e., we find solutions (y, x^i) to every scenario $g^i \in \mathcal{G}$.

As the number of restrictions in $(aSR_2)^{\mathcal{G}}$ grows for larger sets $\mathcal{G}$, the objective value decreases: Let $\mathcal{G}^1 \subseteq \mathcal{G}^2 \subseteq \mathcal{U}_2$ be finite sets. Then the optimal objective value of $(aSR_2)^{\mathcal{G}^1}$ is not smaller than the optimal objective value of $(aSR_2)^{\mathcal{G}^2}$. In fact, as $\mathcal{U}_2$ is the largest set of scenarios that can be considered, the objective value of (aSR_2) is overestimated by this heuristic approach, i.e., for any finite set $\mathcal{G} \subseteq \mathcal{U}_2$, the optimal objective value for $(aSR_2)^{\mathcal{G}}$ is not smaller than the optimal objective value for (aSR_2). On the other hand, we may of course hope for more robustness for larger sets $\mathcal{G}$.

5.1.4.3 Uncertainties $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$

Similar to the discussion in Section 5.1.4.2, the intractability of (aSR_2^Γ) and (aSR_3^Γ) cannot be resolved by using a single, dominating scenario. Instead, the heuristic approach of using a finite subset of scenarios is applicable, and the relations presented at the end of the previous section hold analogously.

5.1.5 Computational Experiments

In this section we report computational results for the robust load planning problem. We evaluate the different concepts according to their nominal quality, i.e. the objective value in case all parameters keep the values that were already assumed for planning, and according to their robustness. To this end, we introduce a measure for robustness in Section 5.1.5.1. Then we analyze the concept of strict robustness by comparing the two uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$ in Section 5.1.5.2 and investigating the effects the parameter Γ has when it is introduced in the uncertainty sets $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$ in Section 5.1.5.3. Finally, we discuss the concept of adjustable robustness in Section 5.1.5.4.

Instances. All reported values are mean values for 20 load planning instances with n between 37 and 85 load units, m from 24 to 38 wagons and an overall number of configurations from 30 to 61 per instance. The test instances are motivated by real-world settings and are described in more detail in [BK12]. The runtimes were restricted to 20 minutes for each instance.

Environment. We used CPLEX 12.2 on a computer with an Intel Core 2 Duo, two cores with 2.2 GHz, 4 GB RAM and Mac OS X 10.7.2.

5.1.5.1 Measuring the Robustness of a Solution

In this subsection we specify how the interval borders of the uncertainty sets are chosen in our experiments. Furthermore, we introduce a robustness measure of a solution and show how it can be computed for the different uncertainty sets.

We define the robustness of a given solution (x, y) as the size of the largest uncertainty set for which it is still strictly robust. To this end, we have to specify the size of our uncertainty sets $\mathcal{U}$. This can be done in different ways. For uncertainty sets $\mathcal{U}_1, \mathcal{U}_2$ and $\mathcal{U}_2^\Gamma$ we assume a percental deviation of σ around the nominal values which specifies the size of the uncertainty sets. For example, for the uncertainty set $\mathcal{U}_1$ and a deviation of σ this means

$$\mathcal{U}_1 = \mathcal{U}_1(\sigma) = \Big\{ u \in \mathbb{R}^n : \max\{0, (1-\sigma)\cdot \tilde{u}_i\} \le u_i \le (1+\sigma)\cdot \tilde{u}_i \Big\}.$$

Due to the maximum-expression for the left border we assure that neither the length nor the weight parameters are negative, this would not make any sense in practice. Analogously, we may write $\mathcal{U}_2(\sigma)$ and $\mathcal{U}_2^\Gamma(\sigma)$ in order to emphasize the dependence of $\mathcal{U}$ on σ. Similarly, we define the size of the uncertainty sets $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$ as Γ, i.e., as the number of elements which may deviate.

In order to determine *how robust* a given solution is we use the following definition which relies on the size of the uncertainty sets measured by a scalar parameter θ. Note that θ may be σ or Γ.

Definition 5.11. *Let a solution (x, y) to an uncertain load planning problem $LP(\xi), \xi \in \mathcal{U}$ be given and assume that the uncertainty set $\mathcal{U} = \mathcal{U}(\theta)$ is dependent on some $\theta \in \mathbb{R}$. Then the robustness of (x, y) is defined as*

$$rob^\theta(x, y) = \max \Big\{ \theta : (x, y) \text{ is feasible for all } \xi \in \mathcal{U}(\theta) \Big\},$$

i.e. $rob^\theta(x, y)$ is the maximal size of the uncertainty set for which (x, y) is still strictly robust.

This definition can be applied to the uncertainty sets $\mathcal{U}_1 = \mathcal{U}_1(\sigma)$, $\mathcal{U}_2 = \mathcal{U}_2(\sigma)$ and $\mathcal{U}_2^\Gamma = \mathcal{U}_2^\Gamma(\sigma)$ resulting in a robustness function $rob^\sigma(x, y)$. Analogously we may apply the definition to $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$ and obtain the robustness function $rob^\Gamma(x, y)$.

In the following we show how to calculate the robustness of a given solution with respect to $\mathcal{U}_1$ and $\mathcal{U}_2$ according to Definition 5.11. For a fixed solution vector $x = (x_1, \ldots, x_n)$ we consider constraints of the type

$$\sum_{j=1}^{n} B_{ij}(\tilde{h}) x_j \le q_i \tag{5.70}$$

for an arbitrary row i where $\tilde{h}$ represents either the nominal overhangs $\tilde{u}$ or the nominal weights $\tilde{g}$. To determine the worst-case that maximizes the left-hand side of (5.70) we introduce

$$\varrho_{ij} := \begin{cases} -1, & \text{if } B_{ij} < 0 \\ 1, & \text{otherwise.} \end{cases}$$

For constraint i let r_i be the maximum percentage by which the values may deviate from the nominal values $\tilde{h}$ such that the constraint is still satisfied:

$$\sum_{j=1}^{n} (1 + \varrho_{ij} \cdot r_i) B_{ij}(\tilde{h}) x_j \leq q_i. \tag{5.71}$$

By solving equation (5.71) for r_i we get:

$$r_i = \frac{q_i - \sum_{j=1}^{n} B_{ij}(\tilde{h}) x_j}{\sum_{j=1}^{n} (\varrho_{ij} \cdot r_j) B_{ij}(\tilde{h}) x_j} \tag{5.72}$$

where the numerator is the slack of the constraint for the nominal values $\tilde{h}$ and the denominator is the sum of the contributions to the left-hand side of the vector $\tilde{h}$.

By taking the minimum of the r_i-values for all constraints i, we obtain the overall robustness of a given solution. Note that we assume a deviation of all elements at the same time. Now we show how we can apply the general robustness equation (5.72) to uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$.

For the uncertainty set $\mathcal{U}_1$ we introduce $r_{js}^{(5.9)}(x, y)$ for all pairs $(j, s) \in \mathcal{M} \times \mathcal{S}_{c(j)}$ of slots s on wagon j with $\sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} > 0$ (i.e. in principle a load unit can be assigned to this slot):

$$r_{js}^{(5.9)}(x, y) = \begin{cases} \infty, & \text{if } \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \geq \sum_{i \in \mathcal{N}} x_{ijs} \cdot \overline{u}_i \\ \frac{\sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} - \sum_{i \in \mathcal{N}} x_{ijs} \cdot \tilde{u}_i}{\sum_{i \in \mathcal{N}} x_{ijs} \cdot \tilde{u}_i}, & \text{otherwise} \end{cases}$$

where for every load unit i the value $\overline{u}_i$ denotes the maximum real-world overhang of the corresponding load unit type. The motivation of the ∞-definition is that we assume that load units always belong to a real-world class. So if a load unit is booked as one with 20 ft corner casting distance, we assume that at most a 26 ft load unit will be delivered because no load units with greater overhang exist in the European setting we analyzed. We assume that if a load unit with 20 ft corner casting distance is assigned to a slot that allows to put a 26 ft load unit on top, there are no problems with feasibility for any scenario. Note that this assumption is not affected by the minimal overhang of the load unit. We do not assume any deviation for trailers because they do not have an overhang in the same sense as containers or swap bodies do. So assignments of trailers are assumed to be feasible for any scenario. Note that the value $r_{js}^{(5.9)}(x, y)$ is also ∞ if no load unit is assigned to slot s on wagon j.

Lemma 5.12. *For the uncertainty set $\mathcal{U}_1$ with parameter σ we have*

$$rob^{\sigma}(x, y) = \min_{(j,s) \in \mathcal{M} \times \mathcal{S}_{c(j)}} r_{js}^{(5.9)}(x, y).$$

For the uncertainty set $\mathcal{U}_2$ we proceed as follows: For constraints (5.38) we introduce the values

$$r_{js}^{(5.38)}(x,y) = \frac{\delta_{c(j),s} - \sum_{i\in\mathcal{N}} \tilde{g}_i \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \tilde{g}_i \cdot x_{ijs}}.$$

for all pairs $(j,s) \in \mathcal{M} \times \mathcal{S}_{c(j)}$ of slots s on wagon j.

For constraints (5.39) we introduce

$$r^{(5.39)}(x,y) = \frac{G - \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_{c(j)}} \tilde{g}_i \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_{c(j)}} \tilde{g}_i \cdot x_{ijs}}.$$

For constraints (5.40) let

$$\varrho_{i\tau s}^{(5.40)} := \begin{cases} -1, & \text{if } d_\tau - 4e_{\tau,s} < 0 \\ 1, & \text{otherwise.} \end{cases}$$

To calculate the robustness w.r.t. constraints (5.40), for every wagon $j \in \mathcal{M}$ we determine the value

$$r_j^{(5.40)}(x,y) = \frac{t_{c(j)} - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \varrho_{i,c(j),s}^{(5.40)} \cdot \tilde{g}_i \cdot \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}.$$

Analogously for constraints (5.41) let

$$\varrho_{i\tau s}^{(5.41)} := \begin{cases} -1, & \text{if } 4e_{\tau,s} - 3d_\tau < 0 \\ 1, & \text{otherwise.} \end{cases}$$

For every wagon $j \in \mathcal{M}$ we define

$$r_j^{(5.41)}(x,y) = \frac{t_{c(j)} - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \tilde{g}_i \cdot \frac{4e_{\tau,s} - 3d_\tau}{d_{c(j)}} \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \varrho_{i,c(j),s}^{(5.41)} \cdot \tilde{g}_i \cdot \frac{4e_{\tau,s} - 3d_\tau}{d_{c(j)}} \cdot x_{ijs}}.$$

For constraints (5.42)-(5.43) we do not have to introduce ϱ-values since all $e_{c(j),s}$ as well as $d_{c(j)} - e_{c(j),s}$ are positive. For every wagon $j \in \mathcal{M}$ let

$$r_j^{(5.42)}(x,y) = \frac{\gamma_{c(j)} - \frac{t_{c(j)}}{2} - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}$$

and

$$r_j^{(5.43)}(x,y) = \frac{\gamma_{c(j)} - \frac{t_{c(j)}}{2} - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \tilde{g}_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{S}_{c(j)}} \tilde{g}_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}.$$

Lemma 5.13. *For the uncertainty set $\mathcal{U}_2$ with parameter σ we have*

$$rob^{\sigma}(x,y) = \min_{(j,s)\in\mathcal{M}\times\mathcal{S}_{c(j)}} \Big\{ r^{(5.38)}_{js}(x,y), r^{(5.39)}(x,y), r^{(5.40)}_{j}(x,y), r^{(5.41)}_{j}(x,y), r^{(5.42)}_{j}(x,y), r^{(5.43)}_{j}(x,y) \Big\}.$$

For uncertainty set $\mathcal{U}_2^{\Gamma}$ and its variant $\overline{\mathcal{U}}_2^{\Gamma}$ we determine the robustness of a solution as follows. For $\mathcal{U}_2^{\Gamma}$ we keep the parameter σ fixed and compute the largest Γ for which a given solution is still feasible. To do this, again we have to consider the robustness according to the different constraints as for uncertainty set $\mathcal{U}_2$ (see above). For each constraint of the form

$$\sum_{j=1}^{n} B_{ij}(g)x_j \le q_i \quad \forall g \in \mathcal{U}_2^{\Gamma} \cup \overline{\mathcal{U}}_2^{\Gamma} \tag{5.73}$$

we choose the vector g^{wc} which is the worst-case in the considered constraint for the uncertainty set $\mathcal{U}_2^{\Gamma}$ or $\overline{\mathcal{U}}_2^{\Gamma}$. According to Lemma 5.5 we have to solve the following MIP to calculate the maximum robustness Γ:

$$\begin{aligned}
&\max \Gamma \\
&\text{s.t.} \ \sum_{j=1}^{n} B_{ij}(\tilde{g})x_j + \Gamma z + \sum_{j=1}^{n} \pi_j \le q_i \\
&\qquad z + \pi_j - (B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j \ge 0 && (j = 1,\dots,n) \\
&\qquad \pi_j \ge 0 && (j = 1,\dots,n) \\
&\qquad z \ge 0
\end{aligned}$$

To determine Γ we do not have to solve the stated MIP, but can determine Γ in a direct way: For each constraint we calculate its left-hand side for the nominal scenario where no element deviates. Afterwards, we compute the change of the left-hand side caused by choosing each load unit individually to be uncertain, which means that the load unit is in the set I. By ordering these changes of the left-hand side in a decreasing order it is easy to compute the robustness rob_i^{Γ} with respect to the analyzed constraint i. We have to add the ordered changes to the left-hand side as long as the constraint stays feasible. The number of values that could be added without violating the constraint i is the robustness rob_i^{Γ}. If we can sum up all changes, rob_i^{Γ} is not limited according to constraint i. The overall robustness is equal to the minimum of the rob_i^{Γ}-values over all constraints i.

Since it is not possible to calculate meaningful strict robust solutions for $\mathcal{U}_3^{\Gamma}$ (because only $x = 0$ is feasible), the robustness of solutions is trivial for uncertainty set $\mathcal{U}_3^{\Gamma}$. For $x = 0$ all load units may deviate (i.e. $rob^{\Gamma}(0,y) = n$), for all other solutions $rob^{\Gamma}(x,y) = 0$.

5.1.5.2 Strict Robustness for the Uncertainty Sets $\mathcal{U}_1$ and $\mathcal{U}_2$

Setup. For each of the 20 test instances we calculated a strict robust solution for varying values of σ using the uncertainties $\mathcal{U}_1$ and $\mathcal{U}_2$. Motivated from real-world settings, we varied σ from 0% to 1025% for $\mathcal{U}_1$, while the range for $\mathcal{U}_2$ is 0% up to 50%. For solutions to (SR_1), we measure the average objective value, the minimal, average, and maximal robustness rob^σ over all solutions that have bounded robustness, and report for how many load unit assignments this is the case. For (SR_2), we report the average objective value, and the minimal, average and maximal robustness rob^σ.

Discussion of results. In Table 5.3 we report computational results w.r.t. $\mathcal{U}_1$. The columns contain the considered deviation σ (in percent), the mean objective value, the minimum, mean and maximum value of rob^σ over all instances (only considered for occupied slots with limited robustness) as well as the percentage of slots for which rob^σ is limited which means that the ∞-case does not apply. These are slots that are neither dedicated to trailers nor allow a maximum overhang which is not smaller than the maximum real-world overhang $\overline{u}_i$ of the assigned load unit.

σ(%)	obj.	min rob^σ	mean rob^σ	max rob^σ	$<\infty$ (%)
0	1845.27	0.00	1.16	23.17	66.85
5	1762.28	6.39	23.42	51.66	47.01
10	1761.59	10.63	23.85	51.66	46.81
15	1752.70	16.78	57.16	532.68	46.29
20	1752.08	21.70	58.43	532.68	45.77
25	1694.96	32.72	74.95	532.68	39.39
35	1694.65	36.05	75.12	532.68	39.32
40	1694.65	43.83	75.55	532.68	39.32
45	1693.38	51.66	78.30	532.68	38.44
55	1642.93	61.52	252.86	769.57	32.52
65	1642.74	71.75	253.89	769.57	32.32
75	1641.57	77.10	256.08	769.57	32.32
80	1634.61	93.66	276.23	769.57	29.89
95	1634.03	118.71	308.67	769.57	29.19
120	1634.03	132.69	310.06	769.57	29.19
135	1632.82	144.93	313.95	769.57	29.19
145	1630.38	198.07	369.88	769.57	28.33
200	1621.21	226.70	620.50	859.51	23.92
230	1620.52	249.52	636.94	859.51	23.72
250	1620.52	271.85	638.05	859.51	23.72
275	1620.16	532.68	662.94	859.51	23.72
535	1615.95	679.02	742.37	859.51	22.94
680	1608.53	769.57	795.89	859.51	22.81
770	1608.53	825.12	832.00	859.51	22.81
830	1554.93	859.51	887.61	1020.49	22.29
860	1470.19	769.57	976.81	1020.49	16.79
885	1457.68	1020.49	1020.49	1020.49	12.10
1025	1439.40	-	-	-	0.00

Table 5.3 Computational results for strict robustness w.r.t. $\mathcal{U}_1$.

There are different greater steps of the objective values e.g. between 0% and 5%, 20% and 25% as well as between 45% and 55%. These greater steps can also be seen in the change of the percentage of load units that have a limited robustness rob^σ. The runtimes are all about 1 second for each instance.

σ(%)	obj.	min rob^σ	mean rob^σ	max rob^σ
0	1845.27	0.00	1.60	6.29
5	1819.19	5.01	6.22	11.77
10	1770.92	10.01	11.15	12.00
15	1684.12	15.05	15.48	17.86
20	1602.32	20.00	22.02	26.67
25	1564.62	25.00	26.15	31.03
30	1478.77	30.12	31.55	35.00
35	1411.70	35.00	36.22	38.18
40	1378.12	40.00	41.74	46.15
45	1344.56	45.00	45.68	46.15
50	1248.34	50.00	50.91	60.14

Table 5.4 Computational results for strict robustness w.r.t. $\mathcal{U}_2$.

The results of strict robustness w.r.t. $\mathcal{U}_2$ are shown in Table 5.4. We report σ, the mean objective values as well as again the minimum, mean and maximum rob^σ. Runtimes are again in the order of a few seconds for each instance.

The results w.r.t. $\mathcal{U}_1$ and $\mathcal{U}_2$ are compared in Figures 5.2, 5.3 and 5.4. In Figure 5.2, the uncertainty parameter σ is plotted against the quotient of the objective value of the resulting strictly robust solution and the nominal objective value (in percent).

We can note a qualitatively different behavior between the solutions to the uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$. While for the latter one (which affects the *weight* of containers) the objective value is smoothly reduced for increasing values of σ, the former (which affects the *overhang* of containers) shows the presence of plateaus in the plot. This is due to the structure of the real-world instances used: Within the considered setting of the European container market, there is only a small set of possible container sizes, while this is not the case for the naturally continuous weights.

A similar behavior can be noted in Figure 5.3, where for each σ the mean objective value of the solutions for the 20 example instances is plotted against its minimum, mean and maximum robustness rob^σ. While in the case of $\mathcal{U}_1$, the robustness of a solution might increase drastically if the objective value is decreased only slightly, this correlation is rather linear for $\mathcal{U}_2$.

Finally, Figure 5.4 shows the robustness of solutions for different values of σ. Note that all values must lie above the diagonal, i.e. have a robustness larger than σ, due to the concept of strict robustness and our definition of robustness. However, the more the robustness of a solution lies above the bisector, the more "implicit" robustness there is. As the plots show, the lines tend to be closer to the bisector in the case of $\mathcal{U}_2$ than for $\mathcal{U}_1$.

We also applied the uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$ simultaneously. This is possible as

$\mathcal{U}_1$ and $\mathcal{U}_2$ affect different constraints. The objective values for different sizes of the uncertainty sets showed a mainly independent behavior for the uncertainty sets. If the size of uncertainty set $\mathcal{U}_1$ is fixed while the size of $\mathcal{U}_2$ is varied, then the behavior is similar to the case where only $\mathcal{U}_2$ is considered – of course, the objective values are slightly worse. This is also the case if the roles of $\mathcal{U}_1$ and $\mathcal{U}_2$ are exchanged.

We conclude from our experiments that studying robustness in detail is more important for uncertainty of type $\mathcal{U}_1$ than for $\mathcal{U}_2$. While for the latter, the correlation between objective value and robustness is rather linear, a large robustness gain at only small costs is possible for the former. Therefore, for $\mathcal{U}_2$ it generally suffices to add a security buffer to the solution, while for $\mathcal{U}_1$ the available data should be carefully considered.

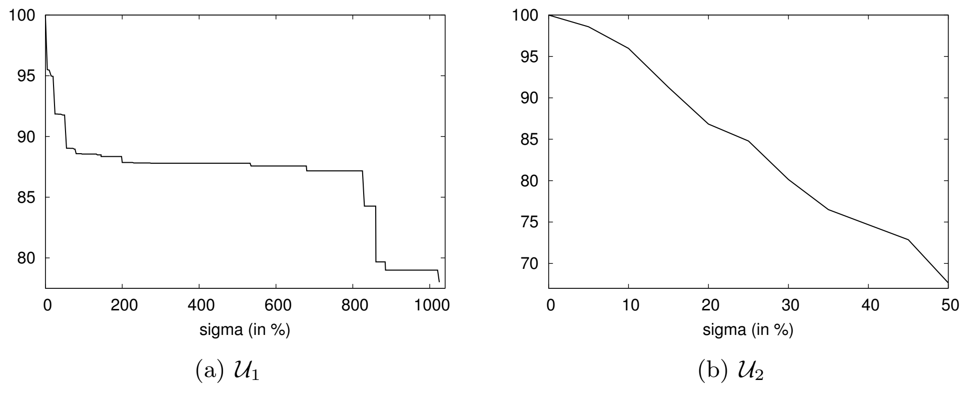


Figure 5.2 Solution quality in percent of nominal objective value for different σ-values.

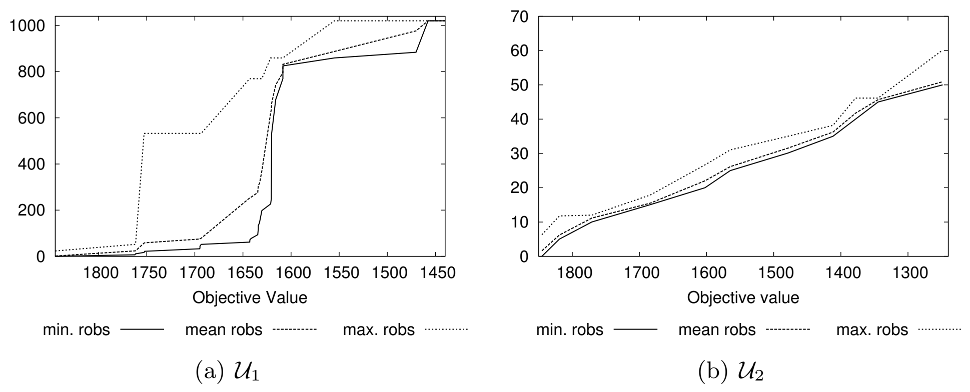
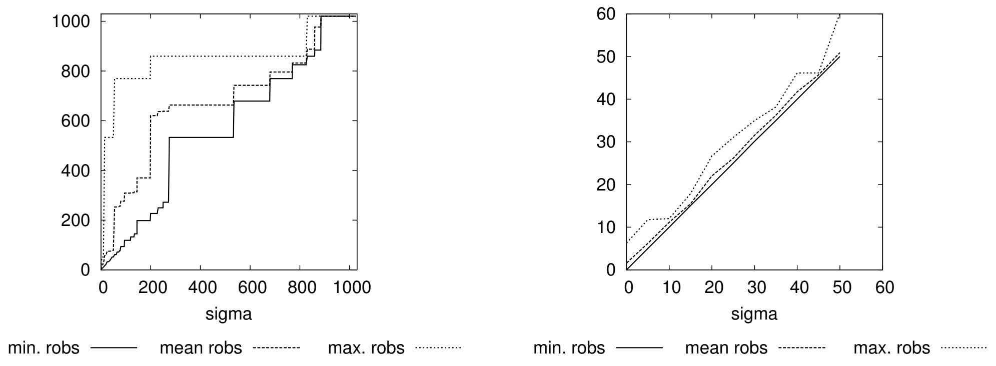


Figure 5.3 Robustness rob^{σ} (in %) for different objective values.



(a) $\mathcal{U}_1$ (b) $\mathcal{U}_2$

Figure 5.4 Robustness rob^{σ} (in %) for different σ-values.

5.1.5.3 The Effects of Γ when Restricting the Uncertainty Set

Setup. In order to analyze the impact of the parameter Γ on strict robust solutions, we solved (SR_2^{Γ}) on the considered benchmark set using $\mathcal{U}_2^{\Gamma}$ with the values $\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\Gamma \in \{0, 1, \dots, 9, 10, 20, \dots, 80, 85\}$, where 85 is the greatest number of load units in one of our example instances. We measured the computation time and determined the respective average values for the objective and rob^{Γ}. Furthermore, we calculated solutions using $\overline{\mathcal{U}}_2^{\Gamma}$ for values $\Gamma \in \{0, 1, \dots, 9, 10, 20, \dots, 80, 85\}$ – note that σ plays no role in this case. We chose $\overline{g}_i$ as the minimum value that is not smaller than g_i and not smaller than 36 tons for load units with a corner casting distance greater than 30 feet and trailers, or 30 tons for load units with 20 feet corner casting distance.

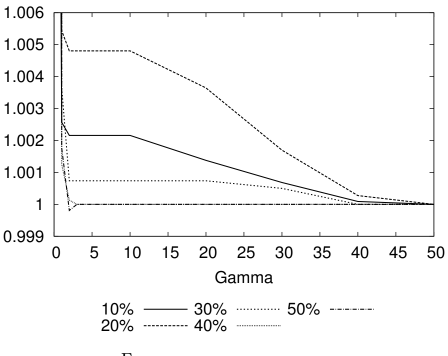
Discussion of results. The results for strict robustness w.r.t. $\mathcal{U}_2^{\Gamma}$ are reported in Table 5.5 for different values of σ. Depending on the size of σ for a sufficiently large number Γ the solutions of $\mathcal{U}_2^{\Gamma}$ equal those of $\mathcal{U}_2$. E.g., for $\sigma = 40\%$ and $\sigma = 50\%$ the value $\Gamma \geq 3$ is large enough. In this situation rob^{Γ} is unbounded (marked by -) because the solution is feasible even if the weights of all load units are uncertain within the intervals defined by σ.

The results for $\overline{\mathcal{U}}_2^{\Gamma}$ are presented in Table 5.6. The instances marked by ' after the number Γ are calculated by the MIP-formulation (SR_2^{Γ}). For all other instances (for $\Gamma \geq 3$) the introduced reformulation is only used for constraints (5.17) which limit the total weight of the train. For constraints (5.21)-(5.24) we do not have to use the reformulation because in these constraints at most three positive x_{ijs} can occur for the example instances used. So there is no difference in using the worst-case formulations stated in constraints (5.21)-(5.24). If for at least one instance the calculation is stopped after the time limit of 20 minutes, we mark the runtime by "*". The runtimes for $\Gamma \leq 3$ and the MIP-formulation (SR_2^{Γ}) are all below 130 seconds with an average value of 60 seconds. On the other hand, if we use the reformulation only for constraints (5.17), for $\Gamma \geq 3$ the runtimes are in order of a few seconds.

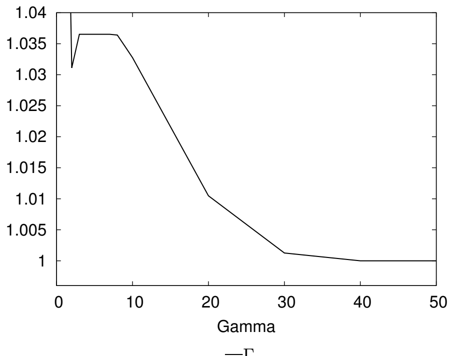
Figures 5.5 and 5.6 illustrate these values. In Figure 5.5, the ratio of the objective value of a solution w.r.t. (SR_2^Γ) to the objective value of the corresponding (SR_2)-solution (which corresponds to $\Gamma = n$) is presented. For increasing values of Γ, both objective values become equal, while the solutions to (SR_2^Γ) are better for small values of Γ. As can be seen in the case of $\mathcal{U}_2^\Gamma$, the objective values are only slightly better than for $\mathcal{U}_2$. This means that the restriction that not all weights deviate from their nominal value at the same time does not significantly increase the nominal quality of a solution. This is different when considering $\overline{\mathcal{U}}_2^\Gamma$, where the increase in the objective value is more significant.

The robustness rob^Γ is presented in Figure 5.6. "Implicit" robustness appears whenever rob^Γ is greater than Γ. For both $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$, the qualitative behavior is roughly the same. For values of Γ larger than 40, the robustness of a solution can be increased at nearly no additional costs. The reason therefore is that for Γ larger than 40 and uncertainty set $\overline{\mathcal{U}}_2^\Gamma$ for our instances only between 27 and 47 (out of the potential 37 to 85) load units can be loaded on the train. Furthermore, the uncertainty set does not affect load units with a nominal weight which is not smaller than $\overline{g}_i$. For $\mathcal{U}_2^\Gamma$ all load unit weights increase if they get uncertain but for small weights the change is very small and sometimes will not affect feasibility of assignments at all. This causes that there is no difference if Γ is increased above 40.

The effects of limiting the number of deviating elements (Γ) showed that it might be worse for practitioners to restrict the number of deviating elements if the deviation is independent of the nominal values (which is the case for $\overline{\mathcal{U}}_2^\Gamma$) and the number of deviating elements is quite small. For $\mathcal{U}_2^\Gamma$ with a deviation around the nominal values there is an increase in the objective values but below 1%.

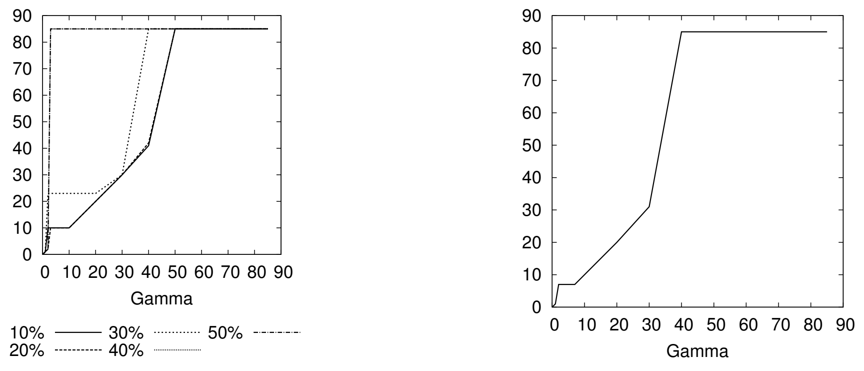


(a) $\mathcal{U}_2^\Gamma$ for different σ-values.



(b) $\overline{\mathcal{U}}_2^\Gamma$

Figure 5.5 Ratio between objective value for limited and unlimited number of deviating elements for different Γ-values.



(a) $\mathcal{U}_2^\Gamma$ for different σ-values. (b) $\overline{\mathcal{U}}_2^\Gamma$

Figure 5.6 Robustness rob^Γ for different Γ-values.

$\sigma(\%)$	10		20		30		40		50	
Γ	rob^Γ	obj.	rob^Γ	obj.	rob^Γ	obj.	rob^Γ	obj.	rob^Γ	obj.
1	1	1775.47	1	1611.03	1	1484.16	1	1379.85	1	1250.55
2	10	1774.74	2	1610.02	23	1479.86	2	1378.30	2	1248.10
3	10	1774.74	10	1610.02	23	1479.86	-	1378.12	-	1248.34
10	10	1774.74	10	1610.02	23	1479.86	-	1378.12	-	1248.34
20	20	1773.36	20	1608.16	23	1479.86	-	1378.12	-	1248.34
30	30	1772.13	30	1605.04	30	1479.51	-	1378.12	-	1248.34
40	41	1771.08	42	1602.76	-	1478.77	-	1378.12	-	1248.34
50	-	1770.92	-	1602.32	-	1478.77	-	1378.12	-	1248.34

Table 5.5 Robustness of strictly robust solutions w.r.t. $\mathcal{U}_2^\Gamma$.

Γ	obj.	rob^Γ	time (sec)
0’	1551.96	0	0.85
1’	1531.76	1	192.07 *
2’	1435.63	7	1200.00 *
3’	1437.66	7	1200.00 *
3	1443.15	7	8.59
10	1437.91	10	243.04 *
20	1406.90	20	245.15 *
30	1394.06	31	545.56 *
40	1392.31	-	423.05 *

Table 5.6 Robustness of strictly robust solutions w.r.t. $\overline{\mathcal{U}}_2^\Gamma$.

5.1.5.4 Discussion of Adjustable Robust Solutions

For the discussion of adjustable robust solutions, there are plenty possible experiments to consider. In this section, we present some of the most interesting results. In some cases, as for the uncertainty set $\mathcal{U}_2$, we found that there is a worst-case scenario in all conducted experiments, and therefore only briefly discuss this in the following.

Setup. In this experiment, we compare adjustable solutions to their strict robust counterpart for $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$, and analyze adjustable solutions for $\mathcal{U}_3^\Gamma$.

We chose $\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\Gamma \in \{0, 1, \ldots, 9, 10, 20, 30, 40\}$. For each parameter pair, and for $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$, we generated 10 initial scenarios using a uniform distribution and used the heuristic approach as described in Section 5.1.4.3 to calculate adjustable solutions. We measured their objective value as the heuristic approximation of the objective function of (aSR_2^Γ), and additionally by the following simulation: We generated 10 independent test-scenarios, completed the adjustable solutions to a solution of each of these scenarios, and measured the objective value of the completed solutions.

For $\mathcal{U}_3^\Gamma$, we used again a set of 10 initial scenarios to calculate heuristic solutions for $\Gamma \in \{0, \ldots, 9\}$, and additional 10 independent test-scenarios to evaluate the objective value after completion to the best possible feasible solution in the respective scenario.

Discussion of results. As already noted at the beginning of this section, for all our benchmark instances we found the existence of a worst-case scenario (see Definition 3.46) in the case of $\mathcal{U}_2$. Though this does not need to be the case from a theoretical point of view, the scenario g_i^{max} dominated all other load weights. Therefore, adjustable and strictly robust solutions are the same.

$\sigma(\%)$	10		20		30		40		50	
Γ	obj.	time	obj.	time	obj.	time	obj.	time	obj.	time
1	1836.75	115	1829.80	135	1820.85	242 *	1816.75	195 *	1811.29	147
2	1833.26	147	1820.91	119	1804.33	214 *	1800.99	228 *	1803.07	262 *
10	1821.89	141	1776.27	240 *	1742.40	300 *	1723.56	433 *	1692.73	326 *
20	1806.47	260 *	1732.59	348 *	1676.92	396 *	1630.98	402 *	1580.07	574 *
30	1799.21	283 *	1687.70	307 *	1599.04	317 *	1536.12	219 *	1450.87	83 *
40	1783.86	234 *	1633.64	270 *	1541.52	384 *	1453.24	253 *	1334.99	120 *
50	1748.46	259 *	1606.87	369 *	1496.17	335 *	1391.83	126	1265.75	116

Table 5.7 Computational results for adjustable robustness w.r.t. $\mathcal{U}_2^\Gamma$.

Table 5.7 shows the mean heuristic objective values of (aSR_2^Γ) and the mean computation times for $\mathcal{U}_2^\Gamma$. In Table 5.8 computational results for the variant $\overline{\mathcal{U}}_2^\Gamma$ are presented. The simulated objective values using the 10 test scenarios are shown in Figure 5.7.

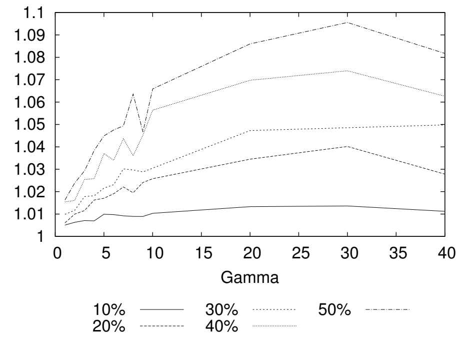
In Table 5.9 we state computational results w.r.t. $\mathcal{U}_3^\Gamma$. The mean runtimes with a time limit of 20 minutes for $\mathcal{U}_3^\Gamma$ and the adjustable heuristic are between 400 and 600 seconds. For all computed Γ for at least one example instance the time limit was

Γ	(SR_2^Γ) obj.	(SR_2^Γ) time	(aSR_2^Γ) obj.	(aSR_2^Γ) time	simulation (SR_2^Γ)	simulation (aSR_2^Γ)
0’	1551.96	0.85	-	-	-	-
1’	1531.76	192.07 *	1842.33	142.92	1717.75	1840.54
2’	1435.63	1200.00 *	1840.37	173.23	1683.34	1836.90
3’	1437.66	1200.00 *	1840.37	173.23	1681.60	1835.47
3	1443.15	8.59	1840.37	173.23	1680.30	1835.47
7	1443.15	7.47	1828.63	520.40	1675.67	1822.81
8	1442.99	4.78	1825.51	485.26	1680.69	1822.49
9	1440.46	242.72 *	1824.06	571.27	1683.50	1817.90
10	1437.91	243.04 *	1818.44	538.23	1680.16	1808.73
20	1406.90	245.15 *	1760.19	554.52	1668.80	1761.58
30	1394.06	545.56 *	1702.21	565.14	1621.53	1689.49
40	1392.31	423.05 *	1638.34	846.25	1562.16	1598.12
all	1392.31	303.51 *	-	-	-	-

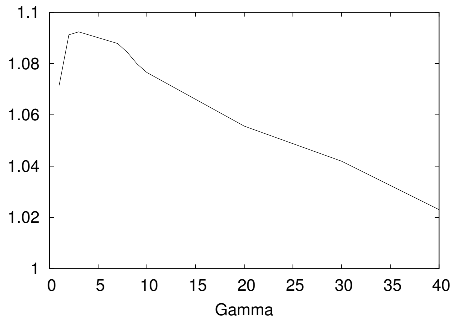
Table 5.8 Computational results for strict and adjustable robustness w.r.t. $\overline{\mathcal{U}}_2^\Gamma$.

Γ	obj.	simA(min)	simA(mean)	simN(min)	simN(mean)	nomA
0	1845.27	-	-	-	-	1845.27
1*	1783.40	1790.53	1806.06	1783.92	1805.59	-
1	1798.85	1789.95	1804.11	1790.61	1805.12	1825.55
2	1758.44	1744.34	1765.16	1740.76	1764.99	1818.56
3	1720.13	1702.58	1724.76	1691.23	1719.75	1811.69
4	1676.80	1658.17	1683.16	1644.94	1674.58	1800.42
5	1636.00	1608.38	1639.34	1600.13	1630.57	1795.36
6	1589.55	1556.37	1594.29	1547.87	1582.48	1789.57
7	1537.91	1510.26	1544.64	1486.57	1532.11	1771.55
8	1495.36	1461.2	1500.27	1441.52	1482.52	1768.66
9	1444.63	1406.11	1445.87	1376.41	1429.66	1751.82

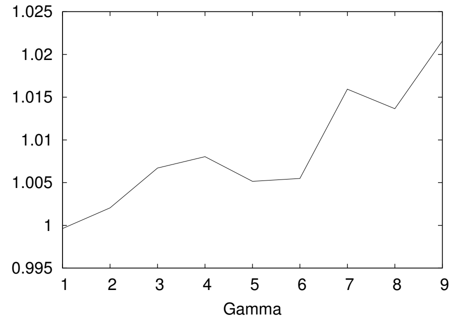
Table 5.9 Computational results for adjustable robustness w.r.t $\mathcal{U}_3^\Gamma$.



(a) $\mathcal{U}_2^\Gamma$ for different σ-values.



(b) $\overline{\mathcal{U}}_2^\Gamma$



(c) $\mathcal{U}_3^\Gamma$

Figure 5.7 Objective value ratio between (aSR_2^Γ) and (SR_2^Γ) for different Γ-values.

reached. The simulated objective values of the adjustable solutions (simA) and the nominal solutions (simN) are given. Furthermore, we computed the mean objective values of the adjustable solutions for the nominal scenarios. In line 1* for the situation where one wagon has a bug, all possible scenarios are considered as initial as well as test scenarios. So the number of scenarios corresponds to the number of wagons. The results show a slightly worse objective value as for the 10 initial scenarios, but there is nearly no difference in the objective value for the simulation, i.e. the number of considered initial scenarios is sufficient (at least for $\Gamma = 1$).

The results of the simulation show that the later decision on the “wait and see”-variables enables much better objective values than the offline optimization that is used with strict robustness. Furthermore, the fixed y_{jk}-values computed with the heuristic for adjustable robustness permit slightly better objective values than the ones obtained by strict robustness (or the nominal problem for $\mathcal{U}_3^\Gamma$).

The objective values for adjustable robustness w.r.t. $\mathcal{U}_2^\Gamma$ (reported in Table 5.7) are better than those of strict robustness (reported in Table 5.5), but also the simulated values are, as Figure 5.7 shows. The same holds for $\overline{\mathcal{U}}_2^\Gamma$, as shown in Table 5.8.

Interestingly, the objective value compared to the strict robust counterpart is different, depending on the uncertainty set used. For $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$, a larger Γ seems to indicate a better (larger) ratio for adjustable robust solutions, while this is the opposite for $\overline{\mathcal{U}}_2^\Gamma$. This can be explained by the considered weights and the flexibility the "here and now"-variables offer. For larger Γ and $\mathcal{U}_2^\Gamma$ there are still load units of different weights which could be assigned to the same slot. So the assignment of load units still offers some flexibility. This is not the case for $\overline{\mathcal{U}}_2^\Gamma$ where for larger Γ nearly all load units have their maximum weights. So the flexibility of changing load unit slot assignments does not enable much better solutions because load unit weights are nearly the same for all load units that fit on a certain slot.

To conclude, using adjustable instead of strictly robust solutions for uncertainty sets $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$ can have a significant impact on the solution quality if in practice the time is available to rearrange the assigned load units. If a practitioner is able to do so, he may increase his gain in terms of the presented objective function by up to 10% in our setting. For uncertainty set $\mathcal{U}_3^\Gamma$ where a limited number of wagons can not be loaded at all it is not so easy to achieve solutions that are much better than the solutions based on the fixed configurations of the nominal solutions. As there is a worse objective value for the nominal solution if no wagon has a bug, for uncertainty set $\mathcal{U}_3^\Gamma$ it may not be sensible to use adjustable solutions if the probability of at least one wagon having a bug is not close to 1.

5.1.6 Concluding Remarks

We introduced the problem of determining load plans in intermodal transportation that take different kinds of uncertainties into consideration. We presented two approaches to include this uncertainty: Strict robust load plans, in which it is assumed that the solution cannot be changed once it is implemented, and adjustable robust load plans, that give the planner the possibility to react once the true problem parameters become known.

For each robustness model and uncertainty set, we analyzed the computational tractability and presented solution approaches. Their performance was analyzed in comprehensive experiments on real-world instances that demonstrated that already at small costs for the planner, the reliability of the solution can be significantly increased.

This suggests that robustness is an important aspect in real-world problems, and can be taken into consideration even for large mixed integer programs with many technical constraints.

We thank Marco Lenk (Deutsche Umschlaggesellschaft Schiene-Straße mbH) for providing information about uncertainties in the practical load planning setting.

5.2 Robust Steiner Trees

5.2.1 Introduction

Having considered approaches to robust counterparts that are based on integer programming formulations in the previous section, we now consider approximation algorithms. To do so, we use one of the classic NP-complete graph problems [Kar72]: The Steiner tree problem (STP).

Definition 5.14. *Let an undirected graph $G = (V, E)$, a weight function $c : E \to \mathbb{R}$ and a set of terminal nodes $T \subseteq V$ be given. Finding a tree $E' \subseteq E$ connecting all terminal nodes, that minimizes the total weight $c(E') = \sum_{e\in E'} c_e$, is called the Steiner tree problem.*

A direct flow-based MIP formulation is the following (see, e.g., [GM93]):

$$\begin{aligned}
\text{(STP)} \quad \min\ & \sum_{e\in E} c_e x_e \\
\text{s.t.}\ & f^h_{ij} + f^k_{ji} \le x_e \quad \forall e = \{i,j\} \in E, h,k \in T \\
& f^k(\delta^+(i)) - f^k(\delta^-(i)) = \begin{cases} 1 & i = r \\ -1 & i = k \\ 0 & i \in V \setminus \{k, r\} \end{cases} \qquad \forall i \in V, k \in T \\
& x \in \mathbb{Z}^{|E|}, f \in \mathbb{R}^{|A|\times|T|}_+,
\end{aligned}$$

where $A = \{(i, j) : \{i, j\} \in E\}$ is the set of directed edges (in both directions), $\delta^+(i)$ denotes all outgoing edges of node i, $\delta^-(i)$ all ingoing edges, and $r \in T$ is an arbitrary root node.

We will write $Steiner(T)$ as the set of all Steiner trees with respect to the terminal nodes T for short; i.e., we rewrite (STP) to

$$\begin{aligned}
\text{(STP)} \quad \min\ & \sum_{e\in E} c_e x_e \\
\text{s.t.}\ & x \in Steiner(T).
\end{aligned}$$

Let α be the approximation ratio of the best known algorithm $\mathfrak{A}$ for (STP). Since 2005, the best-known algorithm achieved a ratio of ≈ 1.55 [RZ05], which has been recently improved to ≈ 1.39 in a yet unreviewed work [BGRS11].

We assume that both the cost vector c and the set of terminals T are uncertain, and consider the following finite uncertainty set of size N:

$$\mathcal{U} = \{(c^1, T^1), \dots, (c^N, T^N)\}$$

We define $T^* := \bigcup_i T^i$, and denote by $(STP(c, T), (c, T) \in \mathcal{U})$ the uncertain Steiner tree problem.

5.2.2 Robust Steiner Trees: The One-Stage Case

We now analyze robust counterparts to the uncertain Steiner tree problem.

In the strictly robust counterpart, we aim at minimizing the worst-case costs of a Steiner tree that is feasible in every scenario. We transform $(STP(c,T),(c,T)\in\mathcal{U})$ to:

$$\begin{aligned}
(\mathrm{SRST}_1)\quad \min \max_{k=1,\dots,N} &\sum_{e\in E} c_e^k x_e\\
\text{s.t.}\quad & x\in \mathit{Steiner}(T^k)\ \forall\ k=1,\dots,N\\
&\Longleftrightarrow\\
\min\ & z\\
\text{s.t.}\quad & z\geq \sum_{e\in E} c_e^k x_e\ \forall\ k=1,\dots,N\\
& x\in \mathit{Steiner}(T^*).
\end{aligned}$$

We now consider some heuristic algorithms for solving $SRST_1$. A natural approach to solve this problem would be to determine the worst-case per edge, and solve the resulting worst-case problem.

Definition 5.15. *Let an uncertain Steiner tree problem* $(STP(c,T),(c,T)\in\mathcal{U})$ *be given. The edge-wise worst-case scenario is then given by* (c^*,T^*)*, where* $c_e^* := \max_{k\in\{1,\dots,N\}} c_e^k$.

Note that in general, (c^*,T^*) is neither an element of $\mathcal{U}$, nor is it a quasi-worst-case scenario.

Heuristic 1 for $\mathbf{SRST_1}$. Solve the edge-wise worst-case scenario using algorithm $\mathfrak{A}$.

The example presented in Figure 5.8 shows that solving the edge-wise worst-case scenario does not necessarily yield an optimal solution to $SRST_1$. Squares are used as terminal nodes. Bold edges denote the Steiner tree with minimal edge costs in each scenario. Here the optimal solution has a worst-case objective value of 8, while Heuristic 1 returns a solution with worst-case objective value of 10. However, we can show that the ratio of worst-case objectives is bounded.

Lemma 5.16. *Heuristic 1 is an αN-approximation algorithm for $SRST_1$.*

Proof. We can proof this lemma analogously to the proof of Proposition 2 from [ABV09]. Let x' be the solution generated by Heuristic 1, and let x^* be an optimal solution of $SRST_1$. Let $\bar{x}$ be the optimal solution to $STP(c^*,T^*)$. Then we have that

$$\max_{k=1,\dots,N} \sum_{e\in E} c_e^k x'_e \leq \sum_{k=1,\dots,N} c_e^* x'_e$$

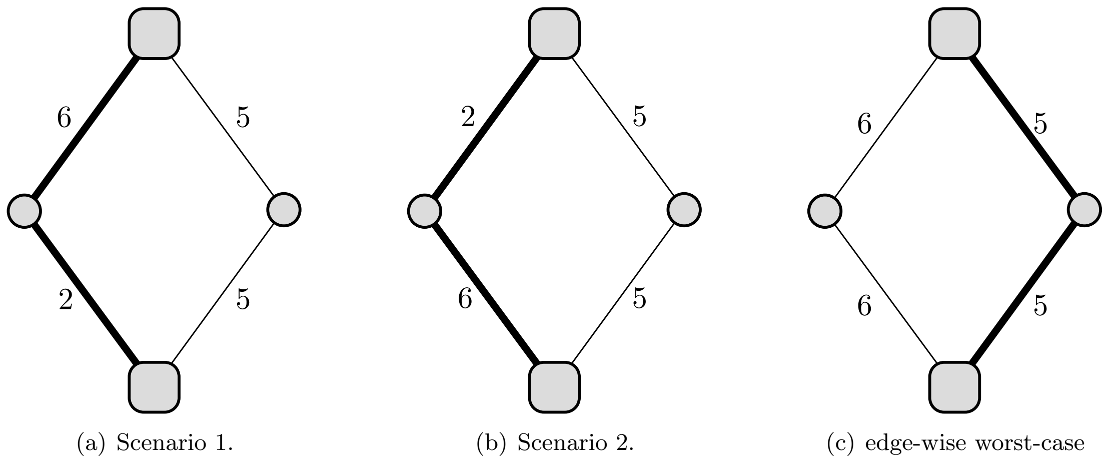


Figure 5.8 Heuristic 1 for $SRST_1$ is not optimal.

$$\begin{aligned}
&\leq \alpha \sum_{e\in E} c_e^* \bar{x}_e \leq \alpha \sum_{e\in E} c_e^* x_e^* \\
&\leq \alpha \sum_{e\in E} \sum_{k=1,\dots,N} c_e^k x_e^* \leq \alpha N \cdot \max_{k=1,\dots,N} c_e^k x_e^*
\end{aligned}$$

□

This bound on the approximation ratio is tight, when the factor α is ignored, i.e., we assume that the edge-wise worst-case scenario is solved to optimality. As an illustration, consider the problem instance in Figure 5.9. On the upper path, there are N edges. For each of these edges, there is a scenario in $\mathcal{U}$ that gives the respective edge the costs M, and ε for all other edges. The costs of the edge on the lower path is always $NM-1$. Now, the optimal solution to the edge-wise worst-case scenario uses the lower path with total costs of $NM-1$ in every scenario, while the upper path has total costs of $M+(N-1)\varepsilon$ in every scenario.

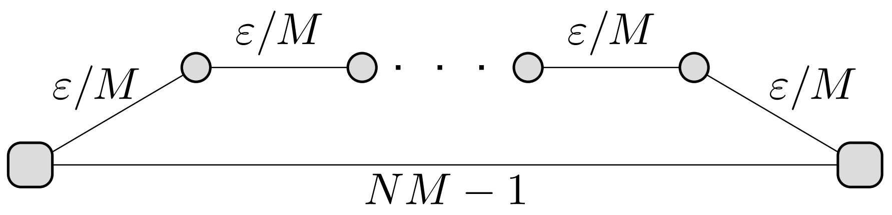


Figure 5.9 The approximation ratio of Lemma 5.16 is tight.

Heuristic 2 for $\mathbf{SRST_1}$. Set $c'_e = \frac{1}{N}\sum_{i=1}^{N} c_e^i$ for every edge, and determine a Steiner tree for this scenario using algorithm $\mathfrak{A}$.

Lemma 5.17. *Heuristic 2 is an αN-approximation algorithm for $SRST_1$.*

Proof. Similar to the case of Heuristic 1, we can use an analogous proof as in [ABV09] for Proposition 1. Let x' be the solution generated by Heuristic 2, and let x^* be the optimal solution to $SRST_1$. Then,

$$\begin{aligned}
\max_{k=1,\dots,N} \sum_{e\in E} c_e^k x' &\leq \sum_{k=1}^{N} \sum_{e\in E} c_e^k x'_e \\
&= N \sum_{k=1}^{N} \frac{1}{N} \sum_{e\in E} c_e^k x'_e \\
&\leq \alpha N \min_{x\in Steiner(T^*)} \sum_{k=1}^{N} c'_e x_e \\
&\leq \alpha N \min_{x\in Steiner(T^*)} \frac{1}{N} \sum_{k=1}^{N} \sum_{e\in E} c_e^k x_e \\
&\leq \alpha N \min_{x\in Steiner(T^*)} \frac{1}{N} N \max_{k=1,\dots,N} \sum_{e\in E} c_e^k x_e \\
&= \alpha N \max_{k=1,\dots,N} \sum_{e\in E} c_e^k x_e^*
\end{aligned}$$

□

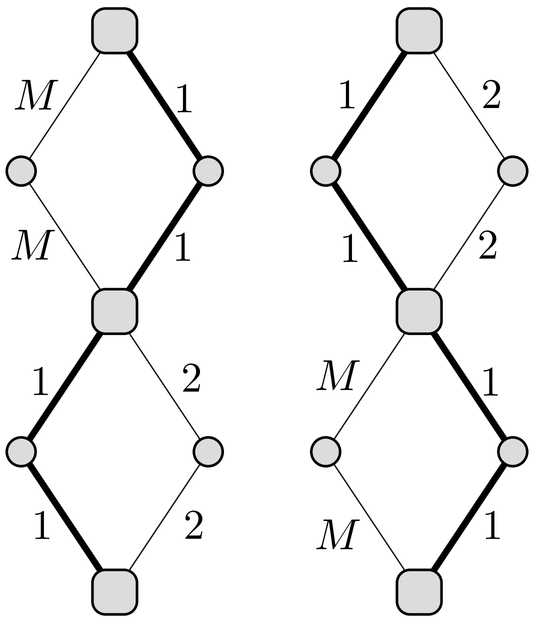
Heuristic 3 for $\mathbf{SRST_1}$. For every scenario, determine an optimal Steiner tree. Among this set of solutions, choose the one with the best worst-case objective value.

Heuristic 3 is not an approximation algorithm – as an example, consider the problem instance in Figure 5.10. Two scenarios with the corresponding optimal Steiner trees are depicted. The worst-case objective value of both trees is $2M + 4$, but there is a solution with a worst-case objective value of 6.

5.2.3 Robust Steiner Trees: The Two-Stage Case

For the two-stage case, we assume the existence of two problem phases: A first phase with incomplete problem information, in which edges can be chosen cheaply, and a second, more expensive phase, in which the full problem information is revealed.

This problem is well-known in the setting of stochastic programming. More specifically, let a finite uncertainty set $\mathcal{U} = \{s^1, \dots, s^N\}$, $|\mathcal{U}| = N$, again be given, where $s^k = (p^k, c^k, T^k)$ and p^k is the probability that scenario k occurs, c^k the edge costs in scenario k, and T^k the terminal nodes in scenario k. In general, there are no further restrictions on c_e^k; however, for approximation algorithms we often assume an inflation



(a) Scenario 1. (b) Scenario 2.

Figure 5.10 Example in which Heuristic 3 can become arbitrarily bad.

factor $\sigma_k > 1$ for each scenario, such that $c_e^k := \sigma_k c_e$. A direct formulation of the two-stage stochastic STP as a MIP is then given by the following program:

$$\begin{aligned} \text{(S-STP)} \quad \min \ & \sum_{e\in E} c_e x_e + \sum_{k=1}^{N} p^k \sum_{e\in E} c_e^k y_e^k \\ \text{s.t.} \ & (x + y^k) \in \mathit{Steiner}(T^k) \ \forall\, k = 1,\dots,N. \end{aligned}$$

In the robust two-stage Steiner tree problem, we do not want to minimize the expected total costs, but the worst-case costs instead. We thus assume that no probabilities p^k are known. As an integer program, we get

$$\begin{aligned} (\text{SRST}_2) \quad \min \ & \sum_{e\in E} c_e x_e + z \\ \text{s.t.} \ & z \geq \sum_{e\in E} c_e^k y_e^k \quad \forall\, k = 1,\dots,N \\ & (x + y^k) \in \mathit{Steiner}(T^k) \ \forall\, k = 1,\dots,N \\ & x \in \mathbb{B}^{|E|},\ y \in \mathbb{B}^{N\cdot|E|},\ z \in \mathbb{R}. \end{aligned}$$

Note that if $c_e^k < c_e, \forall k$ for an $e \in E$, the optimal solution has $x_e = 0$. In [DGRS05], a 15α-approximation algorithm is presented based on a relaxed multi-commodity-flow formulation.

Heuristic 1 for $\mathbf{SRST_2}$. From [DGRS05]: Using an LP-relaxation, we get an approximation ratio of 15α.

We now consider further approximation algorithms.

Heuristic 2 for $\mathbf{SRST_2}$. Set $x = 0$, and solve N separate Steiner tree problems using algorithm $\mathfrak{A}$ to determine y.

Lemma 5.18. *Heuristic 2 for $SRST_2$ is an $\alpha \max_{i\in\{1,\dots,N\}} \sigma^i$-approximation algorithm for $c_e^i = \sigma^i c_e$, $\sigma^i \geq 1$ for all $i \in \{1,\dots,N\}$.*

Proof. Let (x^*, y^*) be an optimal solution to $SRST_2$, and let $(0, \bar{y}^i)$ be an optimal solution in scenario i. Let $A_1,\dots,A_N$ be the costs $A_i = \sum_e \sigma^i c_e \bar{y}_e^i$. As Heuristic 2 solves all scenarios using the algorithm $\mathfrak{A}$, we assume that the algorithm output for scenario i is given by $(0, \hat{y})$ with $\sum_e \sigma^i c_e \hat{y}_e^i = B_i \leq \alpha A_i$.

It holds that

$$f(0,\hat{y}) = \max_{i=1,\dots,N} B_i \leq \alpha \max A_i = \alpha f(0,\hat{y}).$$

Set $j = \arg\max_i A_i$. Then we have

$$\sum_e c_e \bar{y}_e^j \leq f(x^*, y^*),$$

as $(\bar{y}^j, 0)$ is an optimal solution for the STP in scenario j, and (x^*, y^*) has to consider more scenarios. Therefore,

$$\frac{f(0,\hat{y})}{f(x^*,y^*)} \leq \frac{\alpha f(0,\bar{y})}{f(x^*,y^*)} = \alpha\sigma^j \frac{\sum c_e \bar{y}_e^j}{f(x^*,y^*)} \leq \alpha\sigma^j \leq \alpha \max_i \sigma^i$$

which completes the proof. □

Variant 1: Instead of setting $x = 0$, we set $x_e = 1$ for all edges e that are used in all scenario solutions y^i, i.e., in the first phase, we add the intersection of all scenario solutions. As these edges would have been added in every scenario and as $c_e^i \geq c_e$, solutions of Variant 1 of Heuristic 2 never have a worse objective value than solutions of the original heuristic. However, the approximation ratio does not improve.

Variant 2: Add an edge e in the first phase with probability equal to $1/N \sum_{i\in\{1,\dots,N\}} y_e^i$.

Heuristic 3 for $\mathbf{SRST_2}$. Ignore the second phase and use algorithm $\mathfrak{A}$ to determine a Steiner tree with respect to T^* and edge costs c, i.e., set $y^i = 0$ for every scenario.

Lemma 5.19. *Heuristic 3 for $SRST_2$ is an (αN)-approximation for $c_e^i \geq c_e$, if $\bigcap_{i\in\{1,\dots,N\}} T^i \neq \emptyset$.*

Proof. Let $(\bar{x}, 0)$ be the optimal solution for $SRST_2$, when $y^i = 0$ for every scenario i, and let $(\hat{x}, 0)$ be the solution determined by Heuristic 3. Let (x^*, y^*) the optimal solution for $SRST_2$. Then $(x', 0)$ with $x'_e = 1$, if $x^*_e = 1$ or if there is an i with $y^{*i}_e = 1$, is a feasible solution with no added edges in the second phase, if $\bigcap_{i \in \{1,\dots,N\}} T^i \neq \emptyset$. Therefore

$$\sum_{e \in E} c_e \bar{x}_e \leq \sum_{e \in E} c_e x'_e \leq \sum_{e \in E} c_e x^*_e + \sum_{i \in \{1,\dots,N\}} \sum_{e \in E} c^i_e y^{*i}_e.$$

and we have

$$\begin{aligned}
\frac{f(\hat{x},0)}{f(x^*,y^*)} &\leq \alpha \frac{f(\bar{x},0)}{f(x^*,y^*)} \\
&= \alpha \frac{\sum c_e \bar{x}_e}{\sum c_e x^*_e + \max \sum c^i_e y^{*i}_e} \\
&\leq \alpha \frac{\sum c_e \bar{x}_e}{\sum c_e x^*_e + 1/N \sum_i \sum_e c^i_e y^{*i}_e} \\
&\leq \alpha N \frac{\sum c_e \bar{x}_e}{\sum c_e x^*_e + \sum_i \sum_e c^i_e y^{*i}_e} \\
&\leq \alpha N
\end{aligned}$$

which completes the proof. □

The bound in Lemma 5.19 is tight for a fixed α: As an example, consider a star-shaped graph, in which the center node is always a terminal. For every other node, there is exactly one scenario with $c^i = c$ in which this node is a terminal, too. Then, Heuristic 3 adds all edges to the solution, while in an optimal solution only one edge per scenario needs to be added.

5.2.4 Experiments

In this experiment, we evaluate the empirical approximation ratio of the presented one-stage algorithms.

Environment. All experiments were conducted on a PC with 96 GB main memory and an Intel Xeon X5650 processor, running with 6 cores at 2.66 GHz and 12MB cache. All code is written in C++ and has been compiled with g++ 4.4.3 and optimization flag -O3. Mixed-integer programs are solved using Gurobi 5.0 [Gur12] with default parameters.

Instances. We use the "Testset B"-instances from the SteinLib [KMV00], which are of comparably small size. An overview is given in Table 5.10. We chose small-sized problems as we solve the robust counterparts of the instances to optimality.

Instance	$\|V\|$	$\|E\|$	$\|T\|$
b01	50	63	9
b02	50	63	13
b03	50	63	25
b04	50	100	9
b05	50	100	13
b06	50	100	25
b07	75	94	13
b08	75	94	19
b09	75	94	38
b10	75	150	13
b11	75	150	19
b12	75	150	38
b13	100	125	17
b14	100	125	25
b15	100	125	50
b16	100	200	17
b17	100	200	25
b18	100	200	50

Table 5.10 Testset B of SteinLib.

Setup. For every instance, we generate 20 uncertainty sets of two different types: For the uncertainty $\mathcal{U}_1$, we sample $N = 10$ scenarios with edge weights uniformly distributed between $[c_e, 2c_e]$. For the uncertainty $\mathcal{U}_2$, we sample only $N = 5$ scenarios with edge weights that have a 95% chance to be c_e, and a 5% chance to be $5c_e$. For every instance and uncertainty, we solve the nominal problem and the robust counterpart, and calculate the solutions of Heuristics 1, 2, and 3. For all robust approaches, we provide the MIP solver with the nominal Steiner tree as a starting solution. We measure the running time, the nominal objective value, and the worst-case objective value of every solution. The instances b12 and b18 were ignored for our computations, as the nominal solution took too long to be found (>300s).

Results. We first discuss the approximation ratios of Heuristics 1 and 2 under the given setup. We have:

Lemma 5.20. *For a finite uncertainty set $\mathcal{U} \subseteq \times_{e\in E}[c_e, \sigma c_e]$ with $\sigma > 1$ and $|\mathcal{U}| = N$, Heuristic 1 and Heuristic 2 are $\alpha\sigma$-approximations to $SRST_1$.*

Proof. We begin with Heuristic 1. Let x' be the solution generated by Heuristic 1, let $\bar{x}$ be the optimal solution to $STP(c^*)$, and let x^* be the optimal solution to $SRST_1$. Then

$$\max_{k=1,\dots,N} \sum_{e\in E} c_e^k x'_e \le \sum_{e\in E} c_e^* x'_e$$

$$\begin{aligned}
&\leq \alpha \sum_{e\in E} c_e^* \bar{x}_e \leq \alpha \sum_{e\in E} c_e^* x_e^* \\
&\leq \sigma\alpha \sum_{e\in E} c_e x_e^* \leq \sigma\alpha \max_{k=1,\ldots,N} \sum_{e\in E} c_e^k x_e^*
\end{aligned}$$

We now consider Heuristic 2. Let x' be the solution generated by Heuristic 2, and let x^* be the optimal solution to $SRST_1$. Then

$$\begin{aligned}
\max_{k=1,\ldots,N} \sum_{e\in E} c_e^k x_e' &\leq \sigma \sum_{e\in E} c_e x_e' \leq \sigma \sum_{e\in E} c_e' x_e' \\
&\leq \alpha\sigma \min_{x\in Steiner(T^*)} \sum_{e\in E} c_e' x_e \\
&= \alpha\sigma \min_{x\in Steiner(T^*)} \frac{1}{N} \sum_{k=1,\ldots,N} \sum_{e\in E} c_e^k x_e \\
&\leq \alpha\sigma \min_{x\in Steiner(T^*)} \frac{1}{N} N \max_{k=1,\ldots,N} \sum_{e\in E} c_e^k x_e \\
&= \alpha\sigma \max_{k=1,\ldots,N} \sum_{e\in E} c_e^k x_e^*
\end{aligned}$$

□

We now consider the experimental results for $\mathcal{U}_1$ as presented in Tables 5.11, 5.12 and 5.13. Note that the runtimes for the Heuristics 1 and 2 are about the same as in the nominal case, while Heuristic 3 takes even longer than the exact solution. However, this is likely to be due to the small problem size, and the scaling of the runtimes suggests that Heuristic 3 will be faster for larger instances.

In Table 5.12, we compare the respective nominal objective values of the solution. All solution values are close-to-optimal, with only 0.93% deviation for the exact strictly robust solution, and 1.48% for Heuristic 3. Similarly, the worst-case objective values show only slight differences, with the nominal solution being on average 1.04% worse than the exact solution, the Heuristics 1 and 2 slightly better, and Heuristic 3 worse with 2.35%.

The reason for the small differences between the solutions can be found in the structure of the uncertainty set $\mathcal{U}_1$: The higher number of 10 scenarios and the uniform distribution over the interval $[c_e, 2c_e]$ do not penalize heavily when robustness is ignored; in fact, the nominal solution performs quite well in this setting.

However, this picture changes when we consider uncertainties of type $\mathcal{U}_2$. A comparison of runtimes from Figure 5.14 shows that instances are on average more difficult to solve, especially for the exact approach. We mark instances for which at least one of the 20 generated uncertain problems was not solved to optimality within a timelimit of 300s with an asterisk (*).

The structure of $\mathcal{U}_2$ penalizes heavier when uncertainty is neglected, as only some edges become more expensive, but when they do, the difference to the original weights

Instance	Nominal	Exact	Heuristic 1	Heuristic 2	Heuristic 3
b01	0.27	0.52	0.18	0.15	1.50
b02	0.11	0.62	0.27	0.30	3.50
b03	0.47	2.02	0.80	1.07	7.42
b04	0.12	0.52	0.18	0.39	2.04
b05	0.21	2.57	0.66	0.58	5.52
b06	1.82	30.15	2.82	2.61	25.39
b07	0.33	0.87	0.54	0.33	3.99
b08	0.33	1.41	0.71	0.67	8.28
b09	2.52	15.52	3.53	3.37	33.18
b10	0.51	3.85	1.03	0.86	8.79
b11	2.65	15.87	3.01	2.76	28.75
b13	0.82	3.95	1.05	0.95	9.67
b14	1.49	15.98	2.25	1.96	20.80
b15	3.19	11.84	2.76	2.82	27.57
b16	2.19	21.91	3.63	3.72	36.66
b17	3.06	84.55	7.68	6.69	67.70
Average	1.26	13.26	1.94	1.83	18.17

Table 5.11 $\mathcal{U}_1$: Solution time in s.

Instance	Nominal	Exact	Heuristic 1	Heuristic 2	Heuristic 3
b01	0.00	0.98	0.30	0.12	0.79
b02	0.00	0.96	0.30	0.00	1.08
b03	0.00	0.36	0.00	0.00	0.22
b04	0.00	1.19	0.08	0.00	1.53
b05	0.00	0.98	0.33	0.00	1.72
b06	0.00	0.86	0.04	0.04	2.66
b07	0.00	0.68	0.14	0.09	0.99
b08	0.00	0.72	0.05	0.00	0.38
b09	0.00	0.66	0.07	0.07	0.82
b10	0.00	0.35	0.12	0.00	2.56
b11	0.00	1.65	0.23	0.11	1.88
b13	0.00	1.30	0.00	0.00	1.12
b14	0.00	1.02	0.30	0.06	1.96
b15	0.00	0.66	0.09	0.03	1.40
b16	0.00	1.54	0.08	0.00	2.68
b17	0.00	0.99	0.15	0.08	1.91
Average	0.00	0.93	0.14	0.04	1.48

Table 5.12 $\mathcal{U}_1$: Increase of nominal objective value with respect to optimal nominal objective value, in percent.

Instance	Nominal	Exact	Heuristic 1	Heuristic 2	Heuristic 3
b01	0.85	0.00	0.77	0.84	1.09
b02	0.70	0.00	0.61	0.70	1.13
b03	0.25	0.00	0.23	0.46	0.59
b04	0.56	0.00	0.69	0.56	1.89
b05	2.69	0.00	1.74	1.26	4.73
b06	2.16	0.00	1.62	1.99	5.25
b07	0.77	0.00	0.62	0.59	1.36
b08	0.37	0.00	0.34	0.37	0.73
b09	0.91	0.00	0.54	0.64	1.22
b10	0.39	0.00	0.37	0.39	3.26
b11	1.34	0.00	0.81	0.98	2.45
b13	0.52	0.00	0.52	0.52	1.17
b14	0.97	0.00	0.85	1.06	2.83
b15	1.17	0.00	0.81	0.78	2.47
b16	0.92	0.00	0.95	1.03	3.22
b17	2.05	0.00	1.61	1.88	4.21
Average	1.04	0.00	0.82	0.88	2.35

Table 5.13 $\mathcal{U}_1$: Increase of worst-case objective value with respect to optimal worst-case objective value, in percent.

is larger than for $\mathcal{U}_1$. Therefore, nominal and robust solutions differ more, which is reflected in the increased nominal objective values, as can be seen in Table 5.15. Interestingly, solutions generated by Heuristic 3 have very good nominal objective values. This can be explained by the fact that the algorithm considers only one scenario at a time, while the other robust approaches consider all scenarios simultaneously.

Table 5.16 shows that there are large differences in the worst-case objective values of the solutions. For the average row, we ignored those instances that were not solved to optimality in the exact formulation. For instance b15, as an example, both Heuristic 1 and 2 found on average better solutions than the exact approach did within 300s (with small computation times). On average, Heuristic 1 and Heuristic 2 perform only 6.16% or 5.17% worse than the optimal solution, respectively – compared to the approximation guarantee of 500%, a promising result. Both the nominal solution and the solution generated by Heuristic 3 perform significantly worse, with worst-case objective values about 32% higher than the optimal solution.

As a conclusion, we note that the advantage of taking robustness into account depends on the uncertainty under consideration. For uniformly distributed, highly-sampled instances, both the nominal solution and its robust counterpart differ only slightly. For uncertainty sets in which the scenarios may differ significantly, also the robust solution performs very differently to the nominal solution. Furthermore, the better the worst-case costs of a robust solution are compared to the nominal solution, the harder is the MIP formulation to solve, and Heuristics 1 and 2 become attractive.

Instance	Nominal	Exact	Heuristic 1	Heuristic 2	Heuristic 3
b01	0.09	0.16	0.10	0.10	0.50
b02	0.11	0.82	0.20	0.20	1.04
b03	0.47	1.87	0.65	0.63	3.19
b04	0.12	0.25	0.15	0.14	0.79
b05	0.21	0.79	0.52	0.51	2.42
*b06	1.78	149.51	3.05	2.41	14.55
b07	0.15	1.09	0.33	0.29	1.51
b08	0.33	1.93	0.67	0.72	3.34
*b09	2.50	90.87	4.87	4.60	17.23
b10	0.52	6.55	1.01	0.93	4.35
*b11	2.63	141.95	3.40	3.06	16.06
b13	0.62	14.84	0.94	0.81	4.57
*b14	1.50	82.87	2.06	2.01	11.57
*b15	12.32	286.37	15.44	18.30	280.72
*b16	1.87	226.44	3.84	4.13	19.99
*b17	2.88	147.99	14.33	10.37	50.09
Average	1.76	72.14	3.22	3.08	27.00

Table 5.14 $\mathcal{U}_2$: Solution time in s.

Instance	Nominal	Exact	Heuristic 1	Heuristic 2	Heuristic 3
b01	0.00	5.61	2.26	11.46	0.30
b02	0.00	17.17	10.12	26.02	0.84
b03	0.00	6.01	4.75	13.37	0.47
b04	0.00	16.02	11.53	20.85	0.08
b05	0.00	12.21	8.52	17.05	0.00
*b06	0.00	12.54	10.04	19.06	0.74
b07	0.00	9.23	7.43	21.94	1.44
b08	0.00	6.35	7.12	15.96	0.87
*b09	0.00	4.20	4.86	9.14	1.52
b10	0.00	10.87	11.92	20.76	0.47
*b11	0.00	20.11	12.56	27.56	0.57
b13	0.00	10.15	8.24	18.30	1.00
*b14	0.00	5.72	5.81	9.64	0.85
*b15	0.00	2.53	6.16	9.34	1.49
*b16	0.00	12.32	10.12	24.53	1.26
*b17	0.00	7.71	7.71	22.63	1.22
Average	0.00	9.92	8.07	17.98	0.82

Table 5.15 $\mathcal{U}_2$: Increase of nominal objective value with respect to optimal nominal objective value, in percent.

Instance	Nominal	Exact	Heuristic 1	Heuristic 2	Heuristic 3
b01	11.13	0.00	3.80	6.09	11.29
b02	40.82	0.00	7.14	4.05	43.87
b03	15.57	0.00	3.54	5.20	15.65
b04	60.78	0.00	8.25	2.53	63.77
b05	61.57	0.00	6.35	3.47	55.76
*b06	44.00	0.00	6.19	3.53	42.61
b07	18.94	0.00	6.04	6.45	19.97
b08	22.79	0.00	4.96	5.87	20.94
*b09	15.05	0.00	6.02	5.76	17.23
b10	37.59	0.00	5.91	6.21	37.90
*b11	44.76	0.00	6.17	2.87	45.18
b13	26.52	0.00	9.48	6.71	26.12
*b14	23.63	0.00	3.62	5.60	21.31
*b15	10.00	0.00	-8.16	-6.35	11.46
*b16	30.46	0.00	5.58	5.06	31.71
*b17	34.03	0.00	7.15	8.87	32.09
*Average	32.86	0.00	6.16	5.17	32.81

Table 5.16 $\mathcal{U}_2$: Increase of worst-case objective value with respect to worst-case objective value of exact strictly robust program under timelimit, in percent.

5.2.5 Conclusion

We considered approximation algorithms for the one-stage and two-stage robust Steiner tree problem, and presented approximation ratios when possible. In a set of experiments on SteinLib test instances, we evaluated the empirical approximation ratio, which turned out to be significantly better than the theoretical bound. Furthermore, we found that the type of uncertainty set under consideration has a large impact on the performance and importance of robust solutions.

5.3 Robust Periodic Timetabling

5.3.1 Introduction

In Section 4.2 we already discussed how robustness models can be applied to aperiodic timetabling problems that consider a finite time horizon. We now examine the case of *periodic* timetabling problems, in which we assume that train arrivals and departures repeat every T minutes for passenger convencience, see [LM07, Odi96, Nac98, Lie06, Pee03]. The model we use is based on the *Periodic Event Scheduling Problem* (PESP) as introduced in [SU89], where periodically reoccurring events need to be scheduled according to given feasible time spans, minimizing the weighted sum of time differences. Successful applications of this model include the optimized timetable for the underground railway of Berlin [Lie08], and the timetable for the largest Dutch railway contractor, the *Nederlandse Spoorwegen* [KHA^{+}09].

Methods to solve PESP instances most commonly include mixed-integer programming techniques, see [LPW08]. Recently, local search heuristics like the *modulo network simplex* method [NO08, GS11a] have become a valuable alternative due to their ability to tackle larger instances, as it is usually the case in real-world problems.

As delays create considerable passenger inconvenience as well as operational costs and should not be neglected when designing a periodic timetable fit for practice, several attempts have been made to find *robust* timetables, i.e., timetables that behave "well" in a to be specified sense under the existence of delays.

Already the nominal PESP is strongly NP-hard and computationally difficult to solve. Thus, complex robustness models are not a practical option. In [FSZ09], a linear robustness objective is proposed, based on statistical evidence [KDV07]. The resulting model is then solved by a commercial mixed-integer programming solver.

Contribution. In this section, we suggest to fit the robustness concept to the size of the instance, and thus the available computational resources. We consider the application of current robustness models to the timetabling problem, including RecOpt, for which we show that the resulting MIP can be decomposed under certain circumstances. For large-scale instances, we propose the usage of a bicriteria local search algorithm that considers both robustness and travel time in each iteration, approximating the convex hull of efficient solutions. For each robustness approach, we present numerical results.

Overview. In Section 5.3.2, we describe the nominal periodic timetabling problem. We introduce its uncertain formulation in Section 5.3.3 and discuss the application of robustness concepts from the literature, before we consider RecOpt and RecFeas for periodic timetabling in Sections 5.3.4 and 5.3.5. Finally, we discuss a local search algorithm for finding a robust solution in Section 5.3.6, and evaluate our approaches experimentally in Section 5.3.7.

5.3.2 PESP and Periodic Timetabling

A *periodic event* i is a countably infinite set of events i_p, $p \in \mathbb{Z}$, with occurrence times

$$t(i_p) = t(i) + p \cdot T$$

for a given *period* T, see [SU89]. A *span constraint* consists of an interval $[l_{ij}, u_{ij}] \subset \mathbb{R}$ for a pair of events (i, j). The span constraint is satisfied if

$$(t(j) - t(i)) \mod T \in [l_{ij}, u_{ij}].$$

The PESP problem is given as follows: For a given finite set of events with a period T and a finite set of span constraints, find a time $t(i)$ for each periodic event i such that all span constraints are satisfied. It is shown [SU89] that PESP is NP-hard by transformation from the Hamiltonian Circuit Problem.

Based on the PESP, the periodic timetabling problem can be formulated by introducing *event-activity-networks* (EAN) as in Section 4.2 to model the time-dependent behavior of the various vehicles considered [Odi96]. Here the events are periodic in the sense that all arrivals and departures are repeated in every period.

The goal is to find a timetable assigning a *time* $\pi_i := t(i) \mod T \in \mathbb{R}$ to each of the events $i \in \mathcal{E}$ for a given period T such that the span constraints are satisfied, i.e., $(\pi_j - \pi_i) \mod T \in [l_{ij}, u_{ij}]$ for each activity $(i, j) \in \mathcal{A}$. As in the aperiodic case, the objective in the timetabling problem is not only to search for a *feasible* solution, but instead for an *optimal* one, namely we minimize the total passenger traveling time given as

$$\sum_{(i,j)\in\mathcal{A}} (\pi_j - \pi_i) \mod T - l_{ij}.$$

Instead of the event times $\pi_i, i \in \mathcal{E}$, one can equivalently determine the *slack* $y_{ij} = \pi_j - \pi_i - l_{ij}$ for any edge $(i, j) \in \mathcal{A}$ with lower bound l_{ij}. Generally speaking, the slack of an activity is the amount of time spent additionally to its minimum duration. Using this concept, an alternative formulation (used by the modulo network simplex) has been suggested in [Nac98]. Let $\mathcal{T} = (\mathcal{E}, \mathcal{A}_\mathcal{T})$ be a spanning tree with its corresponding fundamental cycle matrix Γ, then the *periodic timetabling problem* can be formulated as follows:

$$\text{(PTT)} \qquad \min \sum_{(i,j)\in\mathcal{A}} \omega_{ij} y_{ij} = TravelTime(y)$$

$$\text{s.t} \quad (y, z) \in \mathcal{F}(l) := \begin{cases} \Gamma(y + l) = Tz \\ 0 \le y_{ij} \le u_{ij} - l_{ij} \ \forall (i, j) \in \mathcal{A} \\ y_{ij} \in \mathbb{R} \ \forall (i, j) \in \mathcal{A} \\ z_{ij} \in \mathbb{Z} \ \forall (i, j) \in \mathcal{A} \setminus \mathcal{A}_\mathcal{T}, \end{cases}$$

where $y = (y_{ij})_{(i,j)\in\mathcal{A}}$ and $l = (l_{ij})_{(i,j)\in\mathcal{A}}$. For details and correctness we refer to [Nac98, Lie06]. As the variables z_{ij} model the periodic character of the problem, they will be referred to as *modulo parameters*.

Note that the modulo parameters are the reason why this problem is NP-hard, and for fixed variables z_{ij} the timetabling problem becomes aperiodic. In this case, it is the dual of a minimum cost flow problem that can be solved efficiently using the network simplex method. On the other hand, the variables z are easily obtained for a given slack vector y. For the sake of simplicity, we will therefore write $y \in \mathcal{F}$, when the values of z are not important in the respective context.

5.3.3 Approaches to Robust Periodic Timetabling

We now briefly consider how some of the generic robustness concepts of Chapter 2 can be applied to the periodic timetabling problem.

Uncertainty Sets We first give an uncertain formulation of the periodic timetabling problem. As explained in Section 4.2, different settings of uncertainty make sense, depending on the real-world situation at hand. We focus on similar uncertainty sets as in the aperiodic case, see Section 4.2:

$$\begin{aligned}\mathcal{U}_1(s) := \{l : \hat{l}_{ij} \le l_{ij} \le (1+s)\hat{l}_{ij}\ \forall(i,j) \in \mathcal{A}^{\text{drive}} \cup \mathcal{A}^{\text{wait}},\\ l_{ij} = \hat{l}_{ij}\ \forall(i,j) \in \mathcal{A}^{\text{change}} \cup \mathcal{A}^{\text{head}}\}\\ \mathcal{U}_2(k,s) := \{l : \hat{l}_{ij} \le l_{ij} \le (1+s)\hat{l}_{ij}\ \forall(i,j) \in D \subseteq \mathcal{A}^{\text{drive}} \cup \mathcal{A}^{\text{wait}},\ |D| = k,\\ l_{ij} = \hat{l}_{ij}\ \forall(i,j) \in \mathcal{A} \setminus D\}\end{aligned}$$

Note that in the periodic setting, these sets have a different meaning than before. They do not describe the delay of a single aperiodic activity (i.e., one train is delayed), but the delay of *every* aperiodic activity that is represented by a periodic activity (i.e., every train of the considered time horizon is delayed). This makes sense when we consider delays that are generated by longer-lasting events, e.g., construction sites or weather conditions.

The uncertain periodic timetabling problem is then given by $(PTT(l), l \in \mathcal{U})$ for $\mathcal{U} = \mathcal{U}_1(s)$ or $\mathcal{U} = \mathcal{U}_2(k,s)$.

Strict Robustness. For $\mathcal{U}_1$, there exists a global worst-case scenario as defined in Section 3.3: The scenario $l^{WC} \in \mathcal{U}_1(s)$ with $l^{WC}_{ij} = (1+s)\hat{l}_{ij}$ for $(i,j) \in \mathcal{A}^{\text{change}} \cup \mathcal{A}^{\text{head}}$, and $l^{WC}_{ij} = \hat{l}_{ij}$ otherwise, dominates all other scenarios of the uncertainty set. Therefore, the strictly robust counterpart of the periodic timetabling problem for $\mathcal{U}_1$ is given by

$$\begin{aligned}(\text{SR}_1) \qquad & \min TravelTime(y)\\ & \text{s.t.}\quad (y,z) \in \mathcal{F}(l^{WC}),\end{aligned}$$

i.e., a problem of type (PTT) again. In the case of $\mathcal{U}_2$, l^{WC} is still a quasi-worst-case scenario, and the problem of finding a strictly robust solution amounts to solving a single scenario as well:

$$(\text{SR}_2) \qquad \min TravelTime(y)$$

$$\text{s.t.} \quad (y,z) \in \mathcal{F}(l^{WC})$$

This is due to the fact that l^{WC} dominates all scenarios from $\mathcal{U}_2$, and although it is not included in $\mathcal{U}_2$, there is always a scenario that would dominate at least one constraint for all vectors $l' \leq l^{WC}$ – for $\mathcal{U}_2$, we have a quasi-worst-case scenario.

Light Robustness. In the lightly robust counterpart, we measure the degree of robustness by how much constraints are violated. As the constraints $\Gamma y = Tz$ stem from the periodicity of the problem and are needed to find a slack vector y that corresponds to a node potential π, we do not allow any violation of these constraints. Instead, we only allow violations of the lower and upper bounds on the activity durations. As possible robust counterpart is then the following:

$$\begin{aligned} \text{(LR)} \qquad \min \ & \sum_{a\in\mathcal{A}} (\gamma_a^1 + \gamma_a^2) \\ \text{s.t.} \ & -\gamma_a^1 \leq y_a \leq u_a - l_a + \gamma_a^2 \ \forall l \in \mathcal{U}, a \in \mathcal{A} \\ & \Gamma(y + \hat{l}) = Tz \\ & \sum_{a\in\mathcal{A}} w_a y_a \leq (1+\rho) f^* \end{aligned}$$

As l^{WC} is a quasi-worst-case scenario, for both $\mathcal{U}_1(s)$ and $\mathcal{U}_2(k,s)$ the robust counterpart is equivalent to the robust counterpart that only considers l^{WC}, i.e.

$$\begin{aligned} \text{(LR)} \qquad \min \ & \sum_{a\in\mathcal{A}} (\gamma_a^1 + \gamma_a^2) \\ \text{s.t.} \ & -\gamma_a^1 \leq y_a \leq u_a - l_a^{WC} + \gamma_a^2 \ \forall a \in \mathcal{A} \\ & \Gamma(y + \hat{l}) = Tz \\ & \sum_{a\in\mathcal{A}} w_a y_a \leq (1+\rho) f^* \end{aligned}$$

Recovery Robustness. In [Sti08], the recovery robust counterpart of the uncertain periodic timetabling for an uncertainty set

$$\mathcal{U}' = \left\{ \quad l \in \mathbb{R}^{|\mathcal{A}|} : \exists a \in \mathcal{A} \text{ s.t } l_{a'} = \begin{cases} \hat{l}_a + \Delta, & \text{if } a' = a \\ \hat{l}_a, & \text{otherwise} \end{cases} \quad \right\}$$

is considered. Note that $\mathcal{U}'$ is finite and consists of the extreme points of an uncertainty set similar to $\mathcal{U}_2(k,s)$. As there is a one-to-one relationship, a scenario $l \in \mathcal{U}'$ can be identified with an activity $a \in \mathcal{A}$. The recovery robust counterpart is then given by the following program.

$$\text{(RR)} \qquad \min_{\pi,f,z} \sum_{(i,j)\in\mathcal{A}} w_{ij}(\pi_j - \pi_i + Tz_{ij})$$

$$\begin{aligned}
\text{s.t.}\quad & \pi_j - \pi_i + f_{ij} + Tz_{ij} = l_{ij}\ \forall (i,j) \in \mathcal{A} \\
& \pi_j - \pi_i + \bar{f}_{ij} + Tz_{ij} = u_{ij}\ \forall (i,j) \in \mathcal{A} \\
\forall a \in \mathcal{U}', \Xi \in \{0,1\} : & \\
& f_{ij} + y^a_j - y^a_i \geq \Delta\chi_{ij}(a)\Xi\ \forall (i,j) \in \mathcal{A} \\
& \bar{f}_{ij} + y^a_i - y^a_j \geq \Delta\chi_{ij}(a)(1-\Xi)\ \forall (i,j) \in \mathcal{A} \\
& D - d^t y^a \geq 0 \\
& y^a \geq 0
\end{aligned}$$

where $\chi_{ij}(a)$ is 1, if $a = (i,j)$, and 0 otherwise.

5.3.4 RecOpt for Periodic Timetabling

We now consider how to apply the RecOpt approach to periodic timetabling.

Distance measure. As specified in Section 3.2, it is crucial to identify a distance measure between two solutions. There are multiple ways to do so (e.g., using the difference of modulo parameters, or graph similarity between spanning tree structures), and we focus on the difference of slack values as a canonical distance measure in this section.

For two periodic timetables $(y^1, z^1), (y^2, z^2)$, we define

$$d\left((y^1, z^1), (y^2, z^2)\right) := \sum_{a \in \mathcal{A}} |y^1_a - y^2_a|$$

MIP formulation. For any given subset $\mathcal{S}$ of scenarios, we consider the following problem:

$$\begin{aligned}
(\text{RecOpt})^{max}(\mathcal{S}) \qquad & \min\ \max_{l \in \mathcal{S}} d\left((y,z),(y^l,z^l)\right) \\
& \text{s.t.}\quad (y,z) \in \mathcal{F}(\hat{l})
\end{aligned}$$

where (y^l, z^l) is an optimal solution to the periodic timetabling problem under scenario l. As noted in Section 3.1.3, $(\text{RecOpt})^{max}(\mathcal{S})$ can be linearized by writing

$$\begin{aligned}
\min\ & c \\
\text{s.t.}\quad & c \geq \sum_{a \in \mathcal{A}} b^l_a\ \forall l \in \mathcal{S} \\
& -b^l_a \leq y_a - y^l_a \leq b^l_a\ \forall l \in \mathcal{S} \\
& (y,z) \in \mathcal{F}(\hat{l})
\end{aligned}$$

where the variables b^l_a are introduced to model the absolute values. This problem can be modified in several ways, depending on the preferences of the planner:

- Instead of minimizing the worst-case recovery distance, the average recovery distance can be minimized, which is denoted by $(\text{RecOpt})^{av}(\mathcal{S})$. The linearization works analogously.
- In the given formulation, the nominal objective is included implicitly via the solutions (y^l, z^l). If a guaranteed nominal quality is desired, a constraint of the form
$$\sum_{a\in\mathcal{A}} w_a y_a \leq (1+\rho) f^*$$
can be added, where f^* denotes the optimal objective value for the nominal problem, and ρ the available budget.

Sampling heuristic. As the presented problem formulation has infinitely many variables for infinite scenario sets $\mathcal{S}$, we cannot solve it directly using $\mathcal{S} = \mathcal{U}_1$ or $\mathcal{U}_2$. Instead we sample a finite number of scenarios and solve a relaxed problem version.

Note that $(\text{RecOpt})^{max}(\mathcal{S})$ is not a PESP, and cannot be solved using the modulo simplex anymore: In general, there is no solution induced by a spanning tree structure that is optimal for $(\text{RecOpt})^{max}(\mathcal{S})$. While the former tend to set slack values to either 0 or $u_a - l_a$, the latter try to add buffer times. However, the modulo simplex can still be used with the objective function of $(\text{RecOpt})^{max}(\mathcal{S})$, although this will slow down the search procedure.

Special problem cases. As most of the presented robustness approaches turn out to be computationally difficult, we consider a special case of the periodic timetabling problem: We assume that $\mathcal{A}_{head} = \emptyset$ and $u_a - l_a = T - 1$ for all $a \in \mathcal{A}_{change}$, and denote this problem as rPESP. This is only slightly restrictive from a practical point of view, but significantly decreases the computational complexity of the problem, as we show in this paragraph.

Lemma 5.21. *Given an rPESP and a vector $d \in \mathbb{N}^{|\mathcal{A}\setminus\mathcal{A}_{change}|}$ with $l_a \leq d_a \leq u_a$ for all $a \in \mathcal{A} \setminus \mathcal{A}_{change}$, there is a feasible solution (y, z) with $y_a = d_a$ for all $a \in \mathcal{A} \setminus \mathcal{A}_{change}$.*

Proof. We construct (y, z) from a node potential π. For every event $e \in \mathcal{E}$ without incoming drive- or wait-activity, set $\pi_e = 0$. All other potentials are then defined using the driving- and waiting times d. As $u_a - l_a = T - 1$ for all change activities a, no lower or upper bound of an activity is violated, and the modulo parameters can be canonically computed from the node potentials. □

Corollary 5.22. *rPESP is combinable (see Definition 3.42) with respect to $y|_{\mathcal{A}\setminus\mathcal{A}_{change}}$.*

Now, using rPESP alone does not make it possible to ignore the set $\mathcal{F}(\hat{l})$ when solving $(\text{RecOpt})^{max}(\mathcal{S})$, i.e., reduce the problem to an unconstrained location problem. This can be seen when we consider a single circle $\mathcal{C}$ and two feasible solutions (y^1, z^1), (y^2, z^2). For any convex combination $\lambda(y^1, z^1) + (1-\lambda)(y^2, z^2)$ with $\lambda \in (0,1)$ it holds

$$\sum_{a \in \mathcal{C}} (\lambda y_a^1 + (1-\lambda) y_a^2) = \lambda T z_{\mathcal{C}}^1 + (1-\lambda) T z_{\mathcal{C}}^2 = T(\lambda z_{\mathcal{C}}^1 + (1-\lambda) z_{\mathcal{C}}^2),$$

which is in general not a multiple of T.

However, when we ignore the duration of change activities in the recovery distance measure, we can solve RecOpt as a sequence of rPESP problems and an unconstrained location problem:

Theorem 5.23. *Let a finite set of scenarios $\mathcal{S} = \{l^1, \dots, l^N\}$ for an rPESP be given. Then, (RecOpt)$^{max}(\mathcal{S})$ and (RecOpt)$^{av}(\mathcal{S})$ with respect to*

$$d_1' \left((y^1, z^1), (y^2, z^2) \right) = \sum_{a \in \mathcal{A} \setminus \mathcal{A}_{change}} \left| y_a^1 - y_a^2 \right|$$

or

$$d_\infty' \left((y^1, z^1), (y^2, z^2) \right) = \max_{a \in \mathcal{A} \setminus \mathcal{A}_{change}} \left| y_a^1 - y_a^2 \right|$$

and (RecOpt)$^{av}(\mathcal{S})$ with respect to

$$d_2'^2 \left((y^1, z^1), (y^2, z^2) \right) = \sum_{a \in \mathcal{A} \setminus \mathcal{A}_{change}} \left(y_a^1 - y_a^2 \right)^2$$

can be solved by solving N rPESP problems and one polynomially solvable location problem, if the optimal solution to every scenario is unique.

Proof. For every scenario l^i in $\mathcal{S}$, we denote the optimal solution of the respective rPESP problem as (y^i, z^i). For d_1' and d_∞', the arising location problems can be formulated as linear programs and are thus polynomially solvable. For $d_2'^2$, setting $y_a = \frac{1}{N} \sum_{i=1}^N y_a^i$ for all $a \in \mathcal{A} \setminus \mathcal{A}_{change}$ yields an optimal solution. Using Lemma 5.21, we are then able to complete these values to a feasible solution (y, z), which minimizes the respective distance by definition. $\square$

In the case of (RecOpt)$^{max}(\mathcal{S})$ with l_∞ distance, the robust timetable can be easily determined by setting $y_a = \frac{1}{2}\left(\max_{i \in \{1,\dots,N\}} y_a^i + \min_{i \in \{1,\dots,N\}} y_a^i\right)$ for every activity $a \in \mathcal{A} \setminus \mathcal{A}_{change}$.

From a practical point of view, the distance measures d' from Theorem 5.23 are justified: In fact, if the recovery distance is supposed to measure the changes in the operating schedule, we are mostly interested in what each driver needs to do differently to the original schedule. Differences in a passenger's change time can therefore be neglected from the operator's point of view.

We now consider the nominal solution quality of a periodic timetable obtained this way.

Theorem 5.24. *For the periodic timetable (y, z) obtained by setting $y_a = \frac{1}{N}\sum_{i=1}^N y_a^i$ for all $a \in \mathcal{A} \setminus \mathcal{A}_{change}$, there holds*

$$TravelTime(y) \leq \frac{1}{N}\sum_{i=1}^{N} TravelTime(y^i) + \sum_{a\in\mathcal{A}_{change}} w_a u_a$$

Proof.

$$\begin{aligned}
TravelTime(y) &= \sum_{a\in\mathcal{A}} w_a y_a \\
&\leq \sum_{a\in\mathcal{A}\setminus\mathcal{A}_{change}} w_a y_a + \sum_{a\in\mathcal{A}_{change}} w_a u_a \\
&= \sum_{a\in\mathcal{A}\setminus\mathcal{A}_{change}} w_a \frac{1}{N}\sum_{i=1}^{N} y_a^i + \sum_{a\in\mathcal{A}_{change}} w_a u_a \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{a\in\mathcal{A}\setminus\mathcal{A}_{change}} w_a y_a^i + \sum_{a\in\mathcal{A}_{change}} w_a u_a \\
&\leq \frac{1}{N}\sum_{i=1}^{N} TravelTime(y^i) + \sum_{a\in\mathcal{A}_{change}} w_a u_a
\end{aligned}$$

□

Note that the proof of Lemma 5.21 constructs a feasible timetable by setting $\pi_e = 0$ for the beginning of each line. The nominal quality of a timetable can be further improved if the starting time of each line is calculated by solving an optimization problem of the form:

$$\begin{aligned}
\min \quad & \sum_{a\in\mathcal{A}_{change}} w_a c_a \\
\text{s.t.} \quad & c_{ij} = x_{l(i)} - x_{l(j)} + d_{ij} + T z_{ij} \ \forall (i,j) \in \mathcal{A}_{change} \\
& c_a \geq l_a \ \forall a \in \mathcal{A}_{change} \\
& 0 \leq x_l \leq T - 1 \ \forall l \in \mathcal{L} \\
& x \in \mathbb{R}^{|\mathcal{L}|}, c \in \mathbb{R}^{|\mathcal{A}_{change}|}, z \in \mathbb{Z}^{|\mathcal{A}_{change}|},
\end{aligned}$$

where $\mathcal{L}$ denotes the set of lines, $l(i)$ denotes the unique line that event $i \in \mathcal{E}$ belongs to, and d_{ij} denotes the time a vehicle needs under the given driving- and waiting-times to arrive at event j, minus the time needed to arrive at event i. The variables x_l denote the offset of the starting time of a line (in the proof of Lemma 5.21, these are set to 0), the variables z_a are modulo parameters, and c_a denotes the change time for change activity a.

Heuristics for this type of problem have been considered in [Hes12]. All quality guarantees that can be found for optimal solutions of this problem can be used to improve the bounds presented in Theorem 5.24

5.3.5 RecFeas for Periodic Timetabling

We now consider the problem of recovering to a feasible solution instead of an optimal one. We use the same distance measure as for RecOpt, i.e. $d\left((y^1,z^1),(y^2,z^2)\right) = \sum_{a\in\mathcal{A}} \left|y^1_a - y^2_a\right|$. For a scenario subset $\mathcal{S}$ of $\mathcal{U}_1$ or $\mathcal{U}_2$, (RecFeas)$^{max}(\mathcal{S})$ is given as

$$\begin{aligned} (\text{RecFeas})^{max}(\mathcal{S}) \qquad & \min \max_{l\in\mathcal{S}} d\left((y,z),(y^l,z^l)\right) \\ & \text{s.t.} \quad (y,z)\in\mathcal{F}(\hat{l}) \\ & \qquad\quad (y^l,z^l)\in\mathcal{F}(l)\ \forall l\in\mathcal{S}, \end{aligned}$$

which can be rewritten to

$$\begin{aligned} & \min\ c \\ \text{s.t.}\quad & c \geq \sum_{a\in\mathcal{A}} b^l_a\ \forall l\in\mathcal{S} \\ & -b^l_a \leq y_a - y^l_a \leq b^l_a\ \forall l\in\mathcal{S} \\ & (y,z)\in\mathcal{F}(\hat{l}) \\ & (y^l,z^l)\in\mathcal{F}(l)\ \forall l\in\mathcal{S} \end{aligned}$$

As it is the case for RecOpt, we may add a constraint that requires a certain nominal quality, or minimize the average recovery distance instead of the maximum distance. Also the sampling heuristic can be applied in this case, and RecFeas can be rewritten to an integer linear program in the case of distance function that comes from a block norm in both the center and median case.

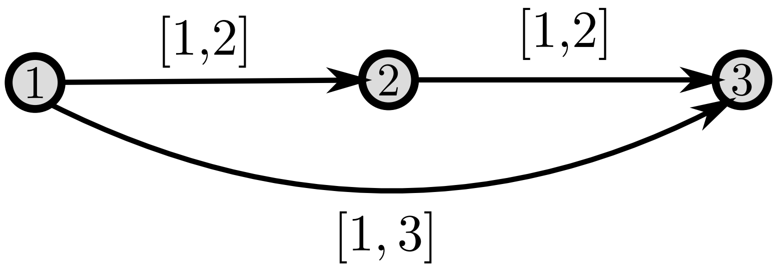
Depending on the uncertainty set, RecFeas might become trivial to solve.

Theorem 5.25. *(RecFeas)$^{max}(\mathcal{U}_1)$ and (SR$_1$) are either both infeasible, or any feasible solution of (SR$_1$) is optimal for (RecFeas)$^{max}(\mathcal{U}_1)$ as well.*

Proof. (RecFeas)$^{max}(\mathcal{U}_1)$ is feasible, iff (PTT)(l) is feasible for all $l\in\mathcal{U}$, which is equivalent to feasibility of (PTT)(l^{WC})=(SR$_1$). If they are feasible, then any solution to (SR$_1$) has 0 recovery costs and is thus optimal for (RecFeas)$^{max}(\mathcal{U}_1)$. □

Note that this is not necessarily the case for $\mathcal{U}_2$, as $l^{WC}\notin\mathcal{U}_2$. In fact, the following example demonstrates that (RecFeas)$^{max}(\mathcal{U}_2)$ might be feasible, when (SR$_2$) is not.

Example 5.26. *Consider the following EAN:*



The brackets next to each activity denote the lower and upper bounds, respectively. The period is $T > 3$, and the uncertainty given by

$$\mathcal{U} = \{(l_{12}, l_{23}) : 1 \leq l_{12}, l_{23} \leq 2,\ l_{12} = 1 \vee l_{23} = 1\}$$

Then every scenario is feasible, and therefore (RecFeas)$^{max}(\mathcal{U})$ as well, but the strict robust counterpart is infeasible.

In our terminology of Section 3.3.2, the reason for this difference between $\mathcal{U}_1$ and $\mathcal{U}_2$ is that in the former case, l^{WC} is a feasibility-worst-case scenario, but only a quasi-worst-case scenario in the latter case.

5.3.6 A Local Search Heuristic

We now assume that our problem instances are too large to be solved to optimality by integer programming methods. Instead, we would like to apply a heuristical strategy with only small computational overhead for robustness purposes. To do so, we calculate solutions that follow a pre-defined buffer distribution.

5.3.6.1 The Modulo Simplex Heuristic for the Periodic Timetabling Problem

We briefly describe the method of [NO08] in this section. Extensions can be found in [GS11a]. Its main idea is to encode a solution as a spanning tree $\mathcal{T}_l \cup \mathcal{T}_u$ by setting the modulo parameters of the tree edges to 0 and the duration of these activities either to their respective lower or upper bound.

Definition 5.27. *[NO08] A* spanning tree structure $(\mathcal{T}_l, \mathcal{T}_u)$ *is a spanning tree* $\mathcal{T} = \mathcal{T}_l \cup \mathcal{T}_u$ *with an edge partition such that* y_{ij} *is set to* 0 *on all edges* $(i, j) \in \mathcal{T}_l$ *and set to* $u_{ij} - l_{ij}$ *for all edges* $(i, j) \in \mathcal{T}_u$.

A spanning tree structure uniquely determines a periodic timetable by calculating the slack y_{ij} for the missing edges $(i, j) \notin \mathcal{T}$ such that the cycle condition $\Gamma(y+l) = Tz$ of (PTT) holds. On the other hand, it is shown in [Nac98] that

$$\binom{\pi}{z} \in \mathcal{Q} := \operatorname{conv}\left\{\binom{\pi}{z} | l_{ij} \leq \pi_j - \pi_i + Tz_{ij} \leq u_{ij}; z \in \mathbb{Z}^m; \pi \in \mathbb{R}^n\right\}$$

is an extreme point of $\mathcal{Q}$ if and only if it is a solution that is given by a spanning tree structure. Thus it is sufficient to investigate only these solutions.

The modulo network simplex works as follows: As it is the case in the classic network simplex method, a given feasible spanning tree solution is gradually improved by exchanging tree and non-tree edges that lie in the same fundamental cycle, i.e., the cycle that consists of the non-tree edge and its unique path in the spanning tree. This is done with the help of a simplex-like tableau.

The objective value $w^t y$ is calculated by $\sum_{(i,j)\notin\mathcal{T}} w_{ij}y_{ij} + \sum_{(i,j)\in\mathcal{T}_u} w_{ij}(u_{ij} - l_{ij})$. Let y^{ij} be the slack vector after pivoting edges e_i and e_j. By writing $[y]_T := y \bmod T$ for short and denoting by b_{ij} the tableau entry for the edges e_i and e_j, the change in the objective value when pivoting a non-tree edge e_i and a tree edge e_j to $\mathcal{T}_l$ is

$$\begin{aligned}
&\omega^t y^{ij} - \omega^t y \\
&= \sum_{k\in\mathcal{A}\setminus(\mathcal{T}\cup\{i\})} \omega_k \left[y_k - \frac{b_{kj}}{b_{ij}} y_i\right]_T + \omega_j \left[\frac{y_i}{b_{ij}} + y_j\right]_T + \sum_{k\in\mathcal{T}_u} \omega_k y_k - \sum_{k\in\mathcal{A}} \omega_k y_k \\
&= \sum_{k\in\mathcal{A}\setminus(\mathcal{T}\cup\{i\})} \omega_k \left(\left[y_k - \frac{b_{kj}}{b_{ij}} y_i\right]_T - y_k\right) + \omega_j \left(\left[\frac{y_i}{b_{ij}} + y_j\right]_T - y_j\right) - \omega_i y_i,
\end{aligned}$$

while the change when pivoting to $\mathcal{T}_u$ is

$$\begin{aligned}
&\Delta\omega_{ij} = \omega_{ij} - \omega \\
&= \sum_{k\in\mathcal{A}\setminus(\mathcal{T}\cup\{i\})} \omega_k \left(\left[y_k - \frac{b_{kj}}{b_{ij}}(y_i - u_i + l_i)\right]_T - y_k\right) \\
&+ \omega_j \left(\left[\frac{y_i - u_i + l_i}{b_{ij}} + y_j\right]_T - y_j\right) - \omega_i(y_i - u_i + l_i).
\end{aligned}$$

Every non-zero entry of the left part of the table stands for a possible basis exchange of a non-tree-arc with a tree-arc that lies on its induced fundamental cycle. However, due to the modulo parameters, reduced costs as in the classic network simplex cannot simply be read off in the tableau. In consequence, the resulting change of every entry of the simplex tableau has to be calculated which results in a time-consuming complexity of $c \cdot (m - n + 1) \cdot (n - 1) \cdot (m - n + 1) = \mathcal{O}(m^2 n + n^3)$, where $m = |\mathcal{A}|$ and $n = |\mathcal{E}|$. Furthermore, as the problem is not convex, many local optima exist, which is the reason why methods of global optimization should be added.

In order to do so, we use the following correspondence between the pivoting operation in the modulo network simplex method and cuts, i.e., sets $\{(i,j) \in \mathcal{A} : i \in \mathcal{E}_1 \text{ and } j \in \mathcal{E}_2\} \cup \{(i,j) \in \mathcal{A} : i \in \mathcal{E}_2 \text{ and } j \in \mathcal{E}_1\}$ for a partition $\mathcal{E}_1 \dot\cup \mathcal{E}_2 = \mathcal{E}$. Every edge e of a spanning tree canonically induces a *fundamental* cut by taking the two connected components that appear when removing e. Pivoting a tree and a non-tree edge as it is done in the modulo network simplex method can therefore be interpreted as shifting slack from the edges of the corresponding fundamental cut to the non-tree edge. Thus, the modulo network simplex searches iteratively for improving fundamental cuts.

Notation. *Let a cut c be given by its node partition $\mathcal{E}_1 \dot\cup \mathcal{E}_2$, and let $\delta \in \mathbb{R}$. We say that we* apply *the cut c with δ, if the slack y_{ij} is increased by δ for all edges (i,j) with $i \in \mathcal{E}_1$, $j \in \mathcal{E}_2$ and decreased by δ for all edges (i,j) with $i \in \mathcal{E}_2$, $j \in \mathcal{E}_1$. Moreover, when the resulting modulo parameters are fixed and a new spanning tree structure is computed, the cut is called* globally improving*, if the objective value decreases.*

To overcome local optima, any other class of cuts can be chosen, which will force the full recomputation of the corresponding simplex-tableau. In that case, the modulo parameters have to be fixed and the dual min-cost balanced flow problem has to be solved in order to obtain a new spanning tree structure.

Figure 5.11 summarizes the main steps of the algorithm. The inner loop and outer loop shown are used for our improvements in the next section.

5.3.6.2 Measuring Robustness

Assuming a robustness function *Robustness* : $\mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$ is given, which evaluates how capable a timetable is to handle delays, we would like to solve the bicriteria robust periodic timetabling problem:

$$\text{(RPTT)} \qquad \begin{cases} \min & TravelTime(y) \\ \max & Robustness(y) \end{cases}$$
$$\text{s.t} \quad (y,z) \in \mathcal{F}$$

There are many possibilities of how to design such a robustness function. One is to generate a large number of random delays, and to measure their impact under a given disposition rule, e.g., postponing every event as far as necessary to become feasible again. However, even though this approach would represent the robustness of a timetable very adequately, it is computationally too difficult to be included in the optimization process.

We will therefore assume that $Robustness(y) = r^t y$ is a linear function, and the weights r_e represent the benefit of buffering edge $e \in \mathcal{A}$. For the following considerations, we use a robustness function that is motivated by the work of [FSZ09] and [KDV07]. Let $pos(e)$ denote the position of a driving or waiting activity e within its corresponding trip $trip(e)$, i.e., the path of the train serving the respective activity within the EAN. We then define the robustness weight r_e of a driving or waiting activity $e \in \mathcal{A}_{drive} \cup \mathcal{A}_{wait}$ as

$$r_e := \frac{\max_{e'}\{\tilde{r}_{e'}\} - \tilde{r}_e}{|\mathcal{A}| \cdot \max_{e'}\{\tilde{r}_{e'}\} - \sum_{e'} \tilde{r}_{e'}}$$
$$\text{where } \tilde{r}_e = (1 - e^{-\lambda \cdot pos(e)}) \cdot (|trip(e)| - pos(e))$$

and the overall robustness of a timetable as

$$Robustness(y) := \sum_{e \in \mathcal{A}_{drive} \cup \mathcal{A}_{wait}} r_e y_e$$

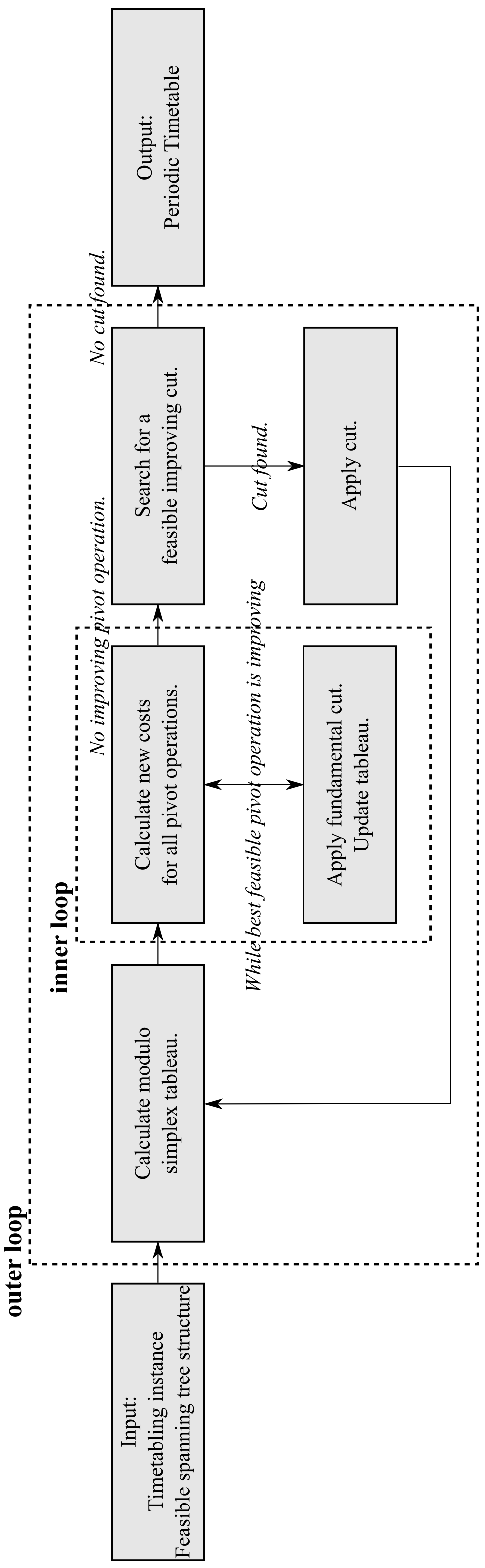

Input:
Timetabling instance
Feasible spanning tree structure
outer loop
Calculate modulo simplex tableau.
inner loop
Calculate new costs for all pivot operations.
While best feasible pivot operation is improving
Apply fundamental cut. Update tableau.
No improving pivot operation.
Search for a feasible improving cut.
Cut found.
Apply cut.
No cut found.
Output:
Periodic Timetable

Figure 5.11 Schematic process of the modulo network simplex.

In preliminary experiments, this robustness function yielded the best results in the sense that timetables that were buffered along this weight distribution turned out to be more robust under random delays than timetables with other tested distributions. However, any other set of robustness weights could be used if it was preferred.

5.3.6.3 Local Search for Robustness

We modify the modulo simplex algorithm in order to find the convex hull of Pareto solutions by the ε-constraint method [Mar67].

Let R be the minimal required robustness of a solution. In order to find a timetable with robustness of at least R, we impose the following two rules on the local search phase:

Rule 1.1: If the current robustness is greater or equal to R, choose a neighbor with robustness greater or equal to R that minimizes the travel time.

Rule 1.2: If the current robustness is smaller than R, choose a neighbor that maximizes the robustness.

Rule 1.1 makes sure that the local search does not violate the robustness constraint, while Rule 1.2 forces the algorithm to try to satisfy the constraint again, should it be violated. Therefore, the only possibility for the robustness constraint to be violated is by means of phase 2.

For phase 2, the following rules hold:

Rule 2.1: If the current robustness is greater or equal to R, minimize the travel time when rebuilding the spanning tree structure from the current modulo parameters.

Rule 2.2: If the current robustness is greater or equal to R, maximize the robustness when rebuilding the spanning tree structure from the current modulo parameters.

Note that Rule 2.2 can only be applied, because the robustness objective *Robustness* is a linear function. All other modifications can still be used for non-linear robustness objectives. A schematic view on the algorithm is given in Figure 5.12. As the algorithm does not necessarily converge, but can in fact endlessly switch between regions with too less and regions with satisfactory robustness, an iteration limit is imposed. When this limit is reached, the best solution found is returned.

We present an example in Figure 5.13, where we plot the travel time and robustness for a single run of the algorithm. The minimal required robustness R is 14. When a local minimum with respect to travel time is found in iteration 476, a connected cut is applied that reduces travel time (Rule 2.1). The algorithm now strives to increase the robustness to a feasible level (Rule 1.2). After failing to do so by getting stuck in a local maximum with respect to robustness, a connected cut is applied that maximizes the robustness again (Rule 2.2.). This continues until the iteration limit of 2000 is reached. The best feasible solution is saved and the algorithm ends.

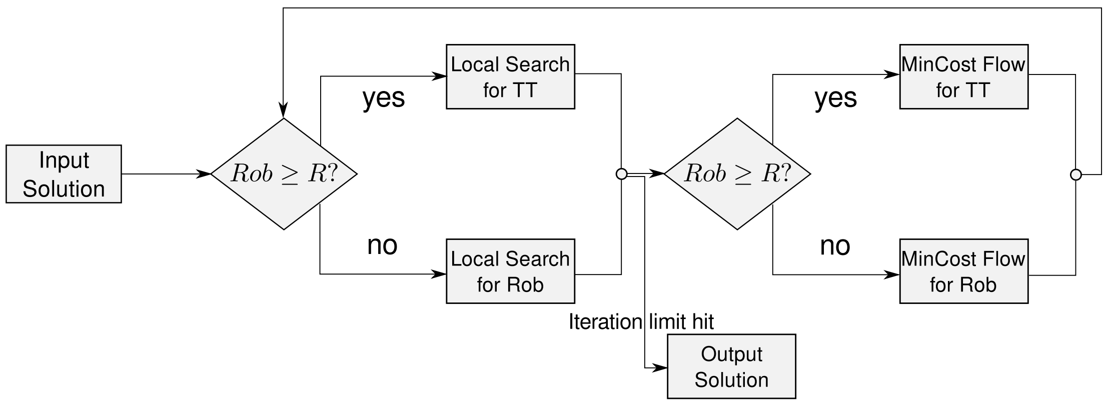


Figure 5.12 Schematic overview of the algorithm.

5.3.7 Experiments

Trying to compare the recovery-to-optimality approach to the local search algorithm from Section 5.3.6 may seem a natural experimental approach. However, we argue that both approaches to robustness do not only differ in the problem size they are designed for, but also in their definition of robustness, and should better be compared to competitors from their own domain. Therefore, we compare the RecOpt approach in Experiment 1 to strict robustness and the nominal solution, and compare the pareto solutions of the local search algorithm to solutions generated using fixed buffer times in Experiment 2. In other words: While the first experiment compares *concepts*, the second experiment compares *algorithms.*

Environment. All experiments were conducted on a PC with 24 GB main memory and an Intel Xeon E5520 processor, running with 4 cores at 2.26 GHz and 8MB cache. Only one core was used and a fraction of the memory was used. All code is written in C++ and has been compiled with g++ 4.4.3 and optimization flag -O3.

5.3.7.1 Experiment 1: Small and Medium Sized Instances

Test instances. We randomly generated two sets of instances: Denote by an instance of size n an instance with n lines of length n, and $1.5n$ change activities. The lower bound l_{ij} for a driving or waiting activity (i, j) is randomly chosen from the interval $[1, 30]$, while the upper bound is chosen from $[2l_{ij}, 2l_{ij} + 40]$. The period T is 60, and change activities have durations from the interval $[3, 62]$. The first set of generated instances consists of instances of size from 5 to 11, 10 each, totalling to 70 instances. The second set consists of instances of size from 20 to 30, 1 each, totalling to 11 instances.

Setup. In this experiment, we try to compare both the direct MIP formulation of RecOpt with respect to d_∞ and the separation approach to the strictly robust and the

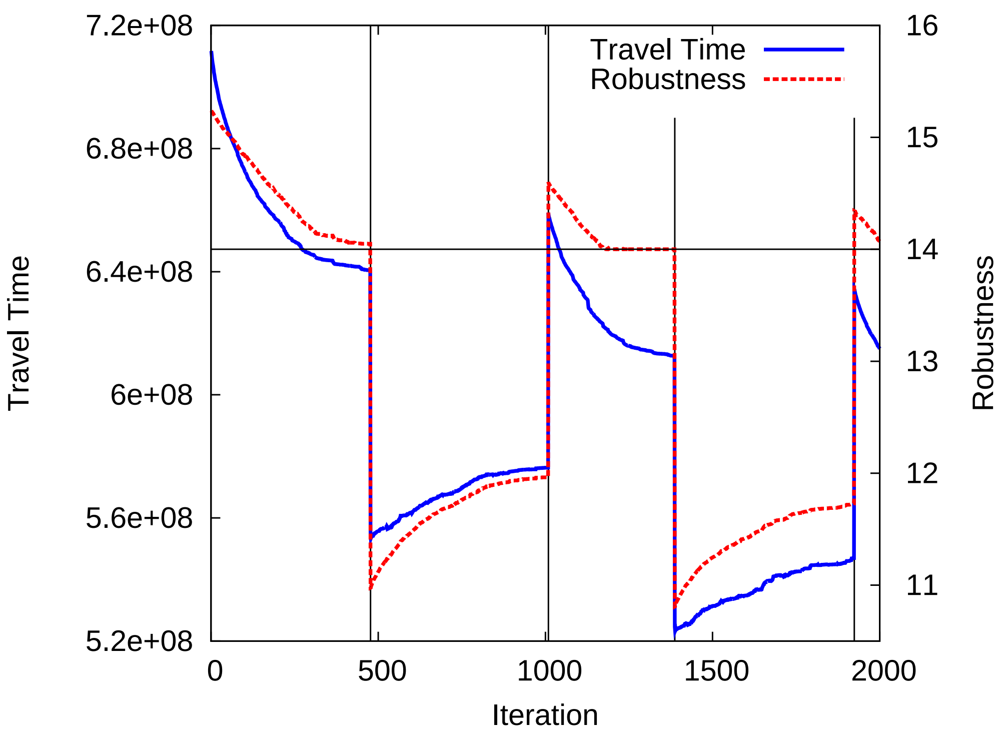


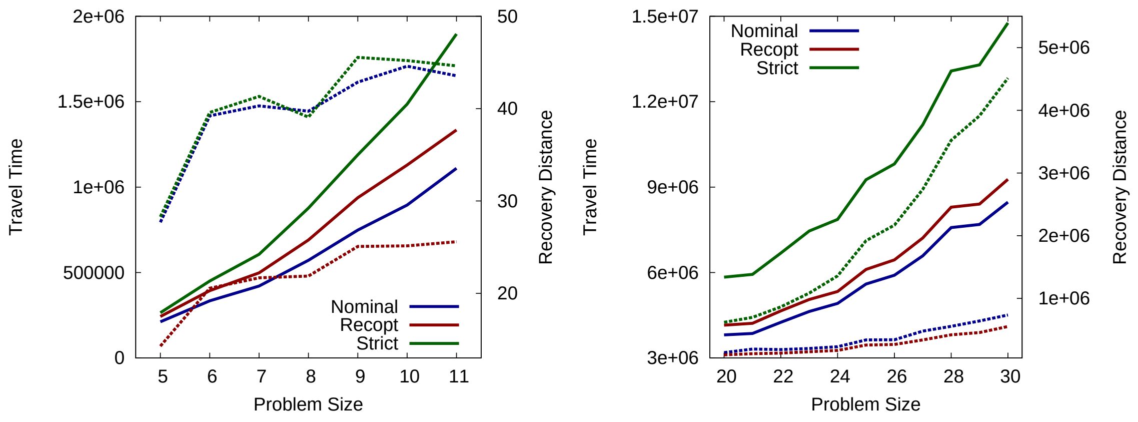
Figure 5.13 An example for the proposed algorithm.

nominal solution. For the smaller dataset we solved RecOpt, (SR) and (PTT) using the MIP-solver Gurobi 5.0 [Gur12]. For the larger dataset exact MIP solutions could not be produced in reasonable time (runtimes > 1h), and the modulo simplex heuristic was used instead. We solved the strictly robust and the nominal problem, and all scenarios seperately. We then calculated the d_2^2 median with respect to driving and waiting activities for all scenarios, fixed these durations, and postoptimized the change durations using Gurobi again, resulting in a heuristic RecOpt solution.

The uncertainty we consider is finite, consisting of $N = |\mathcal{A}|/10$ scenarios. In each scenario, the lower bounds of $|\mathcal{A}|/5$ activities are multiplied by a random factor from $[1.1, 2]$.

Results. Calculation times were in the order of seconds. The results on both datasets are presented in Figure 5.14. While the solid lines represent the value of TravelTime, the dashed lines give the recovery distance. As expected, the travel time of the RecOpt solution is in between the travel time of the strictly robust and the nominal solution. For increasing instance size, the strictly robust solution becomes unattractively expensive, while the objective value of RecOpt keeps close to the nominal objective value.

Concerning the recovery distance, RecOpt yields the smallest costs, as expected. Note that the distance is bounded by $T = 60$ for the smaller instances, as we used the d_∞ center, while the d_2^2 distance is not bounded. While both the strictly robust and



(a) Small instances. (b) Medium instances.

Figure 5.14 Results for experiment 1. Solid lines represent the travel time, dashed lines the recovery radius.

the nominal solution are expensive to recover in the case of the small instances, this is different for the larger instances, where the nominal solution shows a recovery distance that is only slightly larger than the optimum.

The results show that both presented approaches to RecOpt are easily applicable for instances of size up to 30, and suggest that solutions yield a good trade-off between TravelTime and robustness in terms of recovery distance.

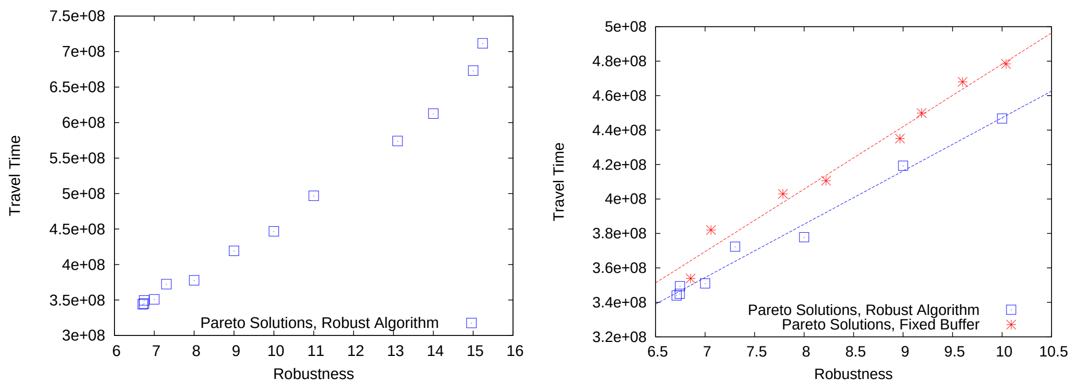
5.3.7.2 Experiment 2: Large Instances

Test instance and starting solution. The high-speed railway instance *bahn-01* from the LinTim-toolbox [GSS] was used, which consists of 3664 events and 6381 activites, of which are 2827 change-, and 3554 waiting- or driving-activities. The network consists of 250 stations and 55 trains. We intended the starting solution to maximize robustness, therefore we fixed the maximally allowed slack for every driving and waiting activity, and minimized the resulting weighted changing times, as this does not decrease the robustness.

Setup. We proceed as follows: As the robustness of the starting solution, i.e., the optimal solution of (RPTT) with respect to *Robustness*, is approximately 15.238, we run the robust local search algorithm with $R \in \{0, 1, \dots, 15\}$, where R is the minimal required robustness, and an iteration limit of 2000. In order to evaluate the efficient solutions found this way, we additionally calculate solutions using the original modulo network simplex method on modified instances, where the robustness level is implemented as a hard constraint on the lower activity bounds. Specifically, we distribute a total slack of $S \in \{0, A, 2A, \dots, 7A\}$ according to the distribution as described in Section 5.3.6, where $A = |\mathcal{A}_{drive} \cup \mathcal{A}_{wait}|$. It is not possible to use more slack than $7A$

this way without violating the upper bounds on the activities. Again, 2000 iterations are used, and the efficient solutions determined.

Results. In Figure 5.15(a), we present the efficient solutions as found by the robust local search. Due to the iteration limit of 2000, robustness levels smaller than 7 did not impose a restriction on the search process. Therefore, all of these runs yield approximately the same solutions, differing only by the randomized connected cuts.



(a) Pareto front as found by the robust local search.

(b) Comparison between pareto front of the robust local search and the fixed buffer approach.

Figure 5.15 Pareto front of solutions.

For the efficient solutions, there seems to be a linear relationship between robustness and travel time, with slope of approximately 1 - meaning that, in order to double the robustness, also passengers need to travel twice as long. Furthermore, the gap between the travel time of the most robust solution, and the travel time of the solution with a robustness level of 15 is larger than would be expected in a linear relationship. This is partially due to the algorithm design, which makes it possible to explore more solutions when the robustness constraint is not tight.

The efficient solutions calculated by ensuring robustness by modifying the lower bounds on the activities are compared to the robust local search solutions of the same robustness level in Figure 5.15(b). Out of 8 solutions, 6 are dominated, while on the other hand, none of the efficient solutions of the robust algorithm is dominated. The least squares fit lines are clearly separated. Table 5.17 shows the respective objective values. The computation time was approximately 1.5 seconds per iteration in all cases, leading to a total runtime of 3000 seconds for every parameter setting.

5.3.8 Conclusion

We considered different approaches to include robustness in the periodic event scheduling problem, proposing to chose the robustness model depending on the problem size and thus available computational resources.

Efficient solutions, rob. algorithm		Efficient solutions, fixed bounds	
Robustness	Travel Time	Robustness	Travel Time
6.712	3.440		
6.743	3.450		
6.746	3.494		
		6.853	3.538
7.000	3.510		
		7.059	3.820
7.303	3.723		
		7.786	4.030
8.000	3.778		
		8.221	4.106
		8.968	4.351
9.000	4.193		
		9.187	4.499
		9.602	4.680
10.000	4.467		
		10.039	4.784
11.000	4.969		
13.105	5.741		
14.000	6.127		
15.000	6.733		
15.238	7.117		

Table 5.17 Objective values of efficient solutions. Travel time values need to be multiplied with 10^8.

For instances where mixed-integer programming can be applied, we presented a model that minimizes the recovery distance to an optimal solution in every scenario, and showed that this problem can be separated depending on the recovery measure. For larger instances, we described a local search algorithm to find periodic timetables with a prescribed robustness level.

In an experimental evaluation, we compared both approaches to competitors for the respective problem size.

5.4 Robust Timetable Information

5.4.1 Introduction

So far, we have considered both the problems of finding a timetable in the periodic and in the aperiodic case, that still performs well under disruptions during operation. We now consider a similar problem from the passenger's point of view: Given a timetable and a set of possible disruptions, what is a good way to travel from origin to destination? We refer to this problem as the *timetable information problem.*

The classic timetable information problem is usually modelled as a shortest path problem in either a time-expanded event-activity network or a time-dependent graph, see [MSWZ07] for a survey. The "reliability" of a path has been considered as an additional search criterion within a multi-criteria timetable information system by [DMS08]. [MS09] and [Sch09] study timetable information in the presence of delays. They show that a massive stream of delay information and schedule changes can be efficiently incorporated into the search for optimal paths. The *robust shortest path problem* has found quite some attention in the literature, see [KY97, BS03]. Uncertainties are modeled by a set of known scenarios, where each scenario corresponds to a set of arc lengths (or weights). The robust shortest path problem is to find among all paths the one that minimizes the path length in the worst case over all scenarios. A related problem is the *robust deviation path problem* which looks for a path such that the maximum difference between the length of this path and the length of the shortest path over all realizations is smallest. [YY98] have shown that the decision versions of both problems are weakly NP-complete, even for layered graphs with only two scenarios. They also give dynamic programming based pseudo-polynomial algorithms for network with a bounded number of scenarios. [ABV05] give an fully-polynomial time approximation scheme for these problems. For an unbounded number of scenarios the problem even becomes strongly NP-hard. Scenarios where arc lengths have an interval range of possible realizations have been studied by [KPY01]. For such scenarios, the robust shortest path problem is trivially solved since the worst case appears when the lengths of all arcs are set to their upper bounds. The robust deviation path problem, however, remains difficult, even for several special cases, see [AL04, Zie04, KZ06a]. [KPY01] also showed that a preprocessing procedure based on the knowledge of those arcs which are never on shortest paths can be used to speed up the solution of the problem. They applied these strategies to acyclic directed graphs in which arc lengths are non-negative intervals. Their experimental study was restricted to graphs with up to 420 nodes which is small compared with the graphs considered here. [MGD04] and [MG05] present exact algorithms for general digraphs. [CLSN11] recently introduced stronger reduction techniques applicable to general digraphs.

In [Büs09] an approximation algorithm for finding a minimal subgraph that contains a shortest path for every scenario is provided in order to facilitate the recovery from a chosen path in case of disturbances.

Our contribution. The classic notion of strict robustness asks to find a solution which is feasible for any scenario. Translated to timetable information this leads to the problem of identifying those transfers which will never break subject to the specified set of delay scenarios. Surprisingly it turns out that already this problem of determining strictly robust changing activities is strongly NP-hard. Due to this hardness result, we use a conservative approximation, i.e. we forbid slightly more changing activities than necessary to guarantee strictly robust solutions. To this end, we compute the maximum amount of delay which can be accumulated for any arrival event. We succeed in developing a dynamic-programming approach for this *delay accumulation problem* which runs in polynomial time for a realistic model of delay scenarios. We also transfer the concept of light robustness to timetable information and develop a solution approach. A lightly robust path is a path which may exceed the minimum travel time in the scenario without any delays (the *nominal scenario*) by not more than a certain specified amount but contains as few as possible changing activities which are not strictly robust under these restrictions. For both concepts we study the price of robustness, originally mentioned in [BS04]: How much longer is the travel time for a robust solution than for a shortest path according to the published schedule? We parametrize the set of considered delay scenarios by the maximum size and number of (large) delays which may occur. Each fixed parameter set can be interpreted as a level of robustness. In computational experiments with the schedule of high-speed trains within Germany of 2011, we explore the trade-off between the level of guaranteed robustness and the increase in travel time for both concepts. Strict robustness turns out to be too conservative, while light robustness is promising: a modest level of guarantees is achievable at a reasonable price for the majority of passengers.

Overview. In Section 5.4.2, we formally introduce event-activity networks as models for timetable information and introduce and discuss delay scenarios. To provide passengers with strictly robust timetable information, that is to find paths that are maintained in every scenario, we need to identify the connections that cannot break. In Section 5.4.3, we study the computational complexity of finding these connections and prove NP-hardness of this problem. Due to this hardness result, we afterwards study the related delay accumulation problem which provides us with a subset of the connections that are always maintained. We derive a dynamic-programming based algorithm to solve this problem. In this way we can solve the NP-hard problem of strictly robust timetable information heuristically in polynomial time. The concepts are extended to light robustness in Section 5.4.5. We present results of our computational study in Section 5.4.6.

5.4.2 Timetable Information and Delay Models

Graph Model. As in Section 4.2, we represent the timetable as an *event-activity network* $\mathcal{N} = (\mathcal{E}, \mathcal{A})$. We briefly repeat its definition: For every arrival and every departure of a train at a station we define an *event*. We also have two virtual events, to be described below. The events $\mathcal{E} = \mathcal{E}_{arr} \cup \mathcal{E}_{dep} \cup \mathcal{E}_{virt}$ are the nodes in the event-activity network. The edges are called *activities*. There are three groups of activities we consider: $\mathcal{A}_{drive}$ contains driving activities of a train between a departure and an arrival event. $\mathcal{A}_{wait}$ contains waiting of a train within a station (i.e. between an arrival and its following departure event), and $\mathcal{A}_{change}$ contains possible changing activities, i.e. transfers between the arrival of a train and the departure of another train (at the same station). Additionally, there are virtual activities $\mathcal{A}_{virt}$ as described below.

As before, the timetable provides a time $\pi_i \in \mathbb{N}$, usually in minutes, for each event $i \in \mathcal{E}$. For each activity $a \in \mathcal{A}$ a length $l_a \in \mathbb{N}$ is given that represents the minimal duration the activity has. A *feasible* timetable satisfies that $\pi_j - \pi_i \geq l_a$ for all $a = (i, j) \in \mathcal{A}$. The *slack time* of an activity $a = (i, j) \in \mathcal{A}$ is defined as $s_a := \pi_j - \pi_i - l_a$. For our timetable information problem we furthermore have a request Req of a passenger. Such a request is specified by an origin station, a destination station and a time $t_{\text{request}} \in \mathbb{N}$ specifying when the passenger can start her journey. In order to model such a request in the event-activity network we add two virtual events, an origin event i_{org} and a destination event i_{dest} and the following set of virtual activities $\mathcal{A}_{virt}$: We connect the origin event to all events $i \in \mathcal{E}_{dep}$ starting at the origin station and having $\pi_i \geq t_{\text{request}}$, and we connect all events $j \in \mathcal{E}_{arr}$ belonging to the destination station and having $\pi_j \geq t_{\text{request}}$ to i_{dest}. If the passenger is interested in a path with earliest arrival time at her destination, we can solve this problem by determining a shortest path from i_{org} to i_{dest} with respect to the following weights

$$c_a := \begin{cases} \pi_j - \pi_i & \text{if } a = (i, j) \in \mathcal{A} \\ \pi_i - t_{\text{request}} & \text{if } a = (i_{org}, i) \in \mathcal{A}_{virt} \\ 0 & \text{if } a = (j, i_{dest}) \in \mathcal{A}_{virt} . \end{cases} \tag{5.74}$$

Summarizing, we denote the nominal timetable information problem as

$$\mathbf{P}(\mathcal{E}, \mathcal{A}, \pi, \text{Req}).$$

The output is a shortest path P^* specified by its sequence of events, and the arrival time at i_{dest} denoted by $f(P^*)$.

Delay scenarios. If everything runs smoothly the passenger would be satisfied with such a shortest path. Unfortunately, delays are unavoidable. This is in particular annoying if a connection on such a path may be missed. The passenger hence may wish to have a reliable connection. To model the uncertainty we define a set of possible exogenous delays, called *source delays*, each of them increasing the lower bound of some activity duration l_a. Examples are obstacles on the tracks that have to be cleared before the train can pass or signalling problems. A scenario is hence given by a vector $d \in \mathbb{N}^{|\mathcal{A}_{wait} \cup \mathcal{A}_{drive}|}$. In real world scenarios one often observes many small source delays, but only a few large ones (which have a direct or indirect effect on a passenger's path). Similar to [BS04], we take this into account and introduce a vector $\epsilon \in \mathbb{N}^{|\mathcal{A}_{wait} \cup \mathcal{A}_{drive}|}$, specifying for each driving or waiting activity a an upper bound ϵ_a for a "small delay".

Moreover, we assume that each source delay is bounded by $d_a^{\max}$ and that the total number of "large" source delays (i.e., those with $d_a > \epsilon_a$) is bounded by K for given values of $d_a^{\max}$ for all $a \in \mathcal{A}$ and an integer K. More precisely, the uncertainty set we consider is given as

$$\mathcal{U} := \mathcal{U}_\epsilon^K := \Big\{ d \in \mathbb{R}^{|\mathcal{A}_{wait} \cup \mathcal{A}_{drive}|} : 0 \le d_a \le d_a^{\max} \text{ for all } a \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}, \\ |\{a \in A : d_a > \epsilon_a\}| \le K \Big\}. \tag{5.75}$$

Delay propagation. When a scenario of source delays $d \in \mathcal{U}$ occurs, it spreads out through the network and results in new times $\pi_i(d)$ for the events $i \in \mathcal{E}$. The basic rule how delays spread along driving, waiting and maintained changing activities is the following: If the start event of an activity $a = (i, j)$ is delayed, also its end event j will be delayed, where the delay can be reduced by the slack time s_a. I.e. we require $\pi(d) \ge \pi$ and

$$\pi_j(d) \ge \pi_i(d) + l_a + d_a \tag{5.76}$$

for all activities $a = (i, j) \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}$. For changing activities equation (5.76) does not necessarily have to hold: If (5.76) holds for a changing activity we say that the connection is *maintained*. If (5.76) does not hold, we say that the connection is *not maintained* or *broken*, meaning that passengers cannot transfer between the corresponding events. This leads to a new set of changing activities which is denoted as $\mathcal{A}_{change}(d)$. We assume that the decision whether a connection should be maintained or not is specified by a fixed waiting time rule: Given a number $wt_a \in \mathbb{N}$ for every changing activity, the connection is maintained if the departing train has to wait at most wt_a minutes compared to its original schedule.

Given these waiting time rules for a given delay scenario d we can propagate the delay through the network along the activities in $\mathcal{A}_{drive} \cup \mathcal{A}_{wait} \cup \mathcal{A}_{change}(d)$, and thus calculate the corresponding adapted timetable according to the following propagation rule:

$$\pi_j(d) = \max \Big\{ \pi_j, \max_{\substack{i:(i,j)\in\mathcal{A} \\ \pi_i(d)+l_{ij} \le \pi_j + wt_{ij}}} \{\pi_i(d) + l_{ij} + d_{ij}\} \Big\} \tag{5.77}$$

where we set $wt_a = \infty \ \forall a \in \mathcal{A}_{wait} \cup \mathcal{A}_{drive}$ and $d_a^{max} = 0 \ \forall a \in \mathcal{A}_{change}$.

For the sake of tractability, this delay propagation model does not take microscopic conflicts like blocked tracks or platforms into account. However, this kind of secondary delays is captured by the small delays ϵ_a which may occur everywhere.

Timetable information under uncertainty. Both $\mathcal{A}$ and π are uncertain parameters for finding the required timetable information since they both depend on the set of source delays d. We hence specify the timetable information problem under uncertainty as

$$\mathbf{P}(\mathcal{E}, \mathcal{A}_{drive}, \mathcal{A}_{wait}, \mathcal{A}_{virt}, \mathcal{A}_{change}(d), \pi(d), \text{Req}), \qquad d \in \mathcal{U}.$$

5.4.3 Strictly Robust Timetable Information

Applied to timetable information the concept of *strict robustness* requires that the path is "feasible" for all delay scenarios, i.e. that all its connections are maintained for any of the scenarios $d \in \mathcal{U}$. The set of strictly robust paths is hence given by

$$\mathbb{SR} = \{P : P \cap \mathcal{A}_{change} \subseteq \bigcap_{d \in \mathcal{U}} \mathcal{A}_{change}(d)\}.$$

In order to determine the set of strictly robust paths, we have to analyze for every changing activity whether it is maintained in all scenarios:

(TT): Transfer-test. Given a changing activity $a = (i, j) \in \mathcal{A}_{change}$, does there exist a delay scenario $d \in \mathcal{U}$ such that a is not maintained?

The set of changing activities that are maintained for all scenarios $d \in \mathcal{U}$ is called the *set of strictly robust activities* and denoted by $\mathcal{A}^{\mathbb{SR}}$. Note that given $\mathcal{A}^{\mathbb{SR}}$, a strictly robust path that has shortest travel time in the nominal case again can be easily computed using a shortest path algorithm in $\mathcal{N}^{SR} = (\mathcal{E}, \mathcal{A}_{wait} \cup \mathcal{A}_{drive} \cup \mathcal{A}^{\mathbb{SR}} \cup \mathcal{A}_{virt})$.

Theorem 5.28. *For the uncertainty set $\mathcal{U}_\epsilon^K$, (TT) is strongly NP-complete, even if $\epsilon_a = 0$ for all $a \in \mathcal{A}$.*

Proof. Note that given a delay scenario d, the resulting delay in all node can be calculated in polynomial time using the delay propagation rule (5.77). Hence, it can be checked in polynomial time, whether a given changing activity a is maintained or not.

The completeness proof is done by reduction from (Minimum-cover), see [GJ79]. An instance of (Minimum-cover) consists of

- a set $M = \{m^1, \ldots, m^n\}$,
- a set of subsets $S = \{s_1, \ldots, s_q\}$ with $s_j \subset M$, and
- a natural number K.

The question to decide is whether there is a subset S' of S with $|S'| \leq K$ such that every $m^i \in M$ is contained in at least one subset $s_j \in S'$. Without loss of generality we assume that for every i, element m^i is contained in at least one set s_j.

Given an instance (M, S, K) of (Minimum-cover), we construct an instance of (TT), consisting of an uncertainty set $\mathcal{U}_0^K$ and an event-activity network $\mathcal{N} = \mathcal{E} \cup \mathcal{A}$. We proceed in two steps: First, we describe the underlying infrastructure, i.e. train paths and stations. Then, we present the resulting event-activity network.

The infrastructure network.

- There is a train tr_{del}, visiting $q + 1$ stations $b_0, \ldots, b_q$.
- At every station b_j for $j = 1, \ldots, q$ for every $i = 1, \ldots, n$, there are trains tr_j^i which for fixed i all run to the same station b^i.

- Furthermore, for every $m^i \in M$ that is contained in the set $s_j \in S$ in the instance of (Minimum-cover), a train $\tilde{tr}_j^i$ runs from b_j to b^i.
- For every $i = 1, \ldots, n$ at station b^i a train tr^i starts and runs to the same station b^*.
- At b^*, a train tr^* starts toward station b^{**}.
- For every $j = 1, \ldots, q$, at station b_j there is one more train $\overline{tr}_j$ starting, which runs to a station b'.
- At b' there is a train tr' departing to tr^*.

An example of this construction for the set $M = \{m^1, m^2\}$ and the set of subsets $S = \{\{m^1, m^2\}, \{m^1\}, \{m^2\}\}$ with $n = 2$ and $q = 3$ is shown in Figure 5.16.

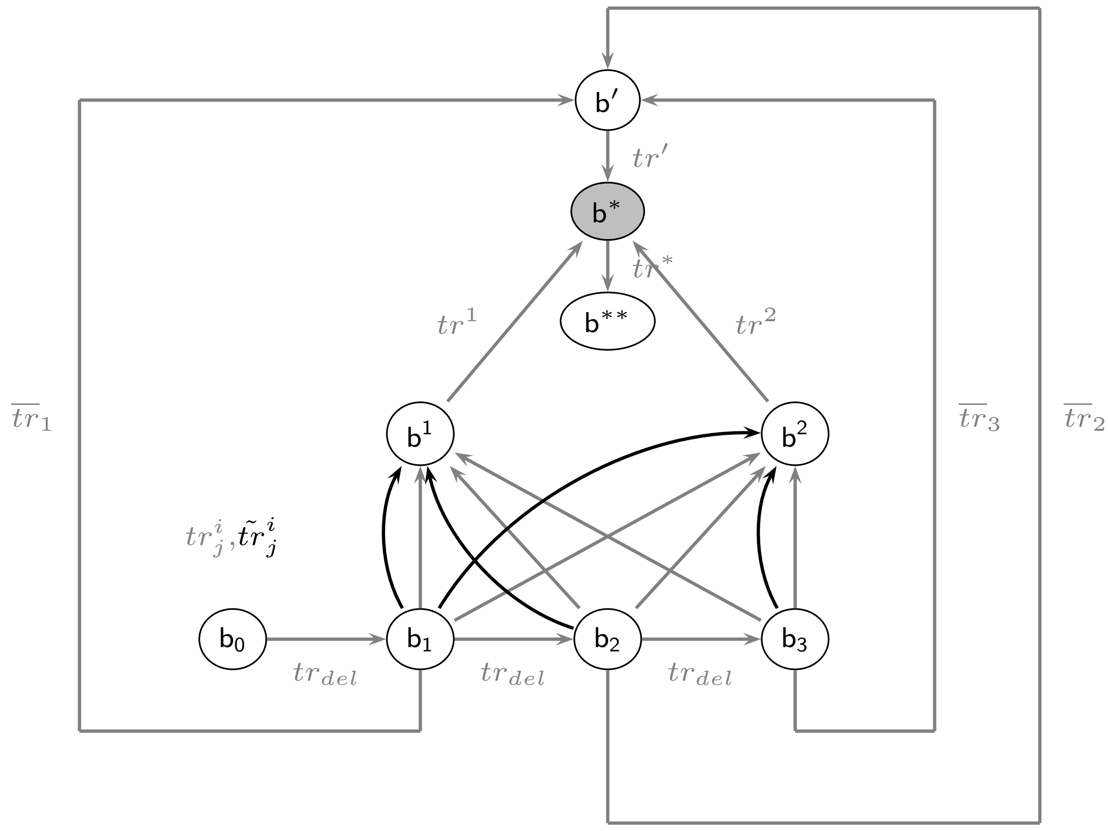


Figure 5.16 Station network showing the reduction of (Minimum-cover) for the proof of NP-completeness of (TT) for $M = \{m^1, m^2\}$ and $S = \{\{m^1, m^2\}, \{m^1\}, \{m^2\}\}$.

We now describe the resulting event-activity network, including its time durations and the set of possible delays.

The event-activity network.

- We assume a slack time and a maximal delay of 0 for all activities for which we do not specify otherwise.
- For tr_{del}, the driving activities have a maximal delay of 2 and the waiting activities a slack of 2. Therefore, any delay occurring on a driving activity of this train will be consumed by the next waiting activity.
- The waiting time of the changing activities between tr_{del} and tr^i_j is set to 2, thus these connections will always be maintained.
- The trains tr^i_j have a slack of 1 on their driving activities.
- At b^i, there is a changing activity between tr^i_j and tr^i with waiting time 1.
- Also at b^i, there is a changing activity between $\tilde{tr}^i_j$ and tr^i with waiting time 2. Hence, if and only if in the current delay scenario there is a delay of $d_{a_j} > 1$ on the edge a_j for a j such that $m^i \in s_j$, a delay bigger than 1 is transferred to the departure of train tr^i at station b^i.
- The waiting time for the changing activity from tr^i to tr^* is 1 for every $i = 1, \ldots, n$. Thus
  1. if in the current delay scenario for every $i = 1, \ldots, n$, there is a j with $d_{a_j} > 1$ for which $m^i \in s_j$, the departure of tr^* is not delayed, and
  2. if the departure of tr^* is not delayed, either 1. holds or there is no delay bigger or equal to 1 in any a_j.
- At b_j, there is a connection from tr_{del} to $\overline{tr}_j$ with waiting time 2.
- The trains $\overline{tr}_j$ have a slack time of 1 on their driving activity.
- The waiting time at b' is 1 for every changing activity from a train $\overline{tr}_j$ to tr'.
- The waiting time for the changing connection from tr' to tr^* is 0.

Therefore, the departure and arrival of tr' is delayed by $\max_j d_{a_j} - 1$ if and only if at least one delay of $d_{a_j} > 1$ occurs on tr_{del}. The corresponding event-activity network to the example shown in Figure 5.16 is given in Figure 5.17.

We now claim that there is a delay scenario $d^* \in \mathcal{U}_0^K$ such that the connection a^* between the trains tr' and tr^* is not maintained, if and only if there is a solution S' to the considered (Minimum-cover) problem.

Suppose there is such an S'. Then we set $d_{a_j} = 2$ for every $s_j \in S'$. Then due to 1. the departure of tr^* is not delayed. But as we saw before, the arrival of tr' in b^* is delayed and because the waiting time of a^* is 0, a^* is not maintained.

Now suppose that there is no such S'. Therefore, due to Condition 2, only for delay scenarios with $d_{a_j} < 1$ for all j, the departure of tr^* is not delayed. But for these delay

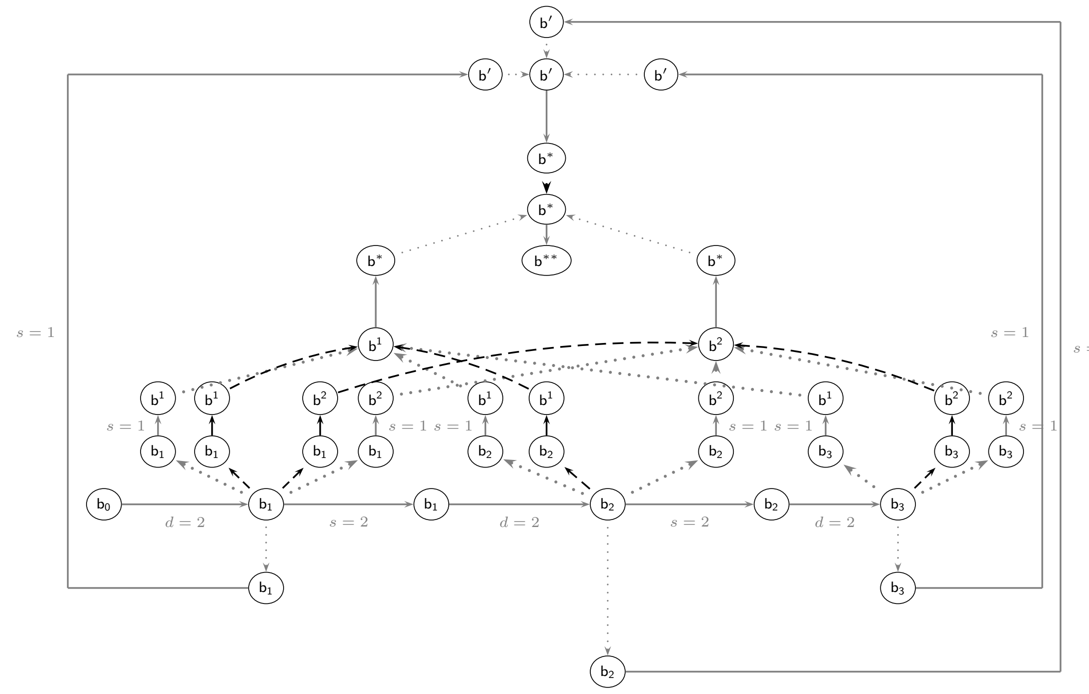


Figure 5.17 Event-activity network with slack times and maximal delays showing the reduction of (Minimum-cover) for the proof of NP-completeness of (TT) for $M = \{m^1, m^2\}$ and $S = \{\{m^1, m^2\}, \{m^1\}, \{m^2\}\}$.

scenarios, because of the slack times on the trains $\overline{tr}_j$, the arrival of tr' is not delayed either, and the connection a^* is maintained. For any other delay scenario $d \in \mathcal{U}_0^K$, tr^* is delayed with delay at least $\max\{0, \max_j d_j - 1\}$. Furthermore, for any delay scenario the arrival of tr' will be delayed by at most $\max\{0, \max_j d_j - 1\}$. Thus connection a^* is maintained for any $d \in \mathcal{U}_0^K$. $\square$

Note that this proof also holds when we assume the delays to be in $\mathbb{N}_0$.

An intuitive explanation why transfer test is computationally hard is the following: A changing activity a can either be maintained *actively* by holding a train in a station for at most wt_a minutes, or *passively* if the train that is entered has delay of its own due to other reasons, i.e., whether a changing activity $a = (i, j)$ is maintained or not depends on the time values $\pi_i(d)$ *and* $\pi_j(d)$. Both values may or may not be influenced by the same source delay of some earlier event. So the core difficulty is to decide whether there is *no* delay scenario that simultaneously delays event i by a certain amount but does not delay event j too much.

We are not aware of any reasonable way to solve (TT) exactly. Still, we can calculate a subset of the strictly robust connections using the following observation: Let $a = (i, j)$ be a changing activity. Then, if $\pi_i(d) \le \pi_i + s_a + wt_a$ for all $d \in \mathcal{U}$, a is maintained for every delay scenario and hence $a \in \mathcal{A}^{SR}$. Thus the set of connections $\mathcal{A}^{acc}$ having this property is a subset of the strictly robust connections. Then, every path in the network $\mathcal{N}(\mathcal{A}^{acc}) = (\mathcal{E}, \mathcal{A}_{drive} \cup \mathcal{A}_{wait} \cup \mathcal{A}^{acc} \cup \mathcal{A}_{virt})$ that contains only connections from $\mathcal{A}^{acc}$ is a strictly robust path. Note that to check the above-mentioned property, we only have to check whether the delay in i can exceed $s_a + wt_a$ or not. As we do not have to mind the consequences of the delay in j, this problem turns out to be much easier than (TT) as we will see in Section 5.4.4.

Slightly generalizing, this leads to the related problem:

(DA): Delay accumulation. Given an event $j^* \in \mathcal{E}$, an uncertainty set $\mathcal{U}$ and a number $D \in \mathbb{N}$, does there exist a delay scenario $d \in \mathcal{U}$ such that $\pi_{j^*}(d) = \pi_{j^*} + D$?

We now explain how to apply (DA). Consider again a changing activity $a = (j^*, i) \in \mathcal{A}_{change}$. If we solve (DA) for the event j^* and corresponding $D = s_a + wt_a + 1$, then the answer “no” proves that $a \in \mathcal{A}^{SR}$. This is sufficient because of the following monotonicity property: If there is a delay scenario which accumulates a delay of D at some event j^*, then it is also possible to generate every smaller delay at j^* (the latter is a consequence from Lemma 5.32 below). Hence, solving (DA) for every $a = (j^*, i) \in \mathcal{A}_{change}$ and corresponding $D = s_a + wt_a + 1$, we obtain a subset of the strictly robust connections $\mathcal{A}^{acc} \subset \mathcal{A}^{SR}$. Every path in the network $\mathcal{N}(\mathcal{A}^{acc})$ that contains only connections from $\mathcal{A}^{acc}$ is a strictly robust path. Thus given the network $\mathcal{N}(\mathcal{A}^{acc})$, we can solve the strictly robust timetable information problem heuristically in polynomial time. A small observation that might be of theoretical interest is the following: (DA) is equivalent to (TT) if the underlying undirected graph of the event-activity network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ is acyclic or if $wt_a = 0$ holds for all $a \in \mathcal{A}_{change}$.

5.4.4 Efficiently Solving Delay Accumulation

In the following we assume that the delays are natural numbers. This is a no restriction in timetable instances because for all practical purposes time will discretized in time steps (minutes, fractions of minutes, or seconds).

We show how problem (DA) can be solved in polynomial time. To this end we derive properties of the delays that allow us to restrict our search to only a subset of delay scenarios when solving (DA). Due to this result we are able to develop Algorithm 5 that solves (DA) in polynomial time and can hence be used to determine $\mathcal{A}^{acc}$. As before, we consider an event-activity network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ and delay scenarios d on $\mathcal{N}$. For an event $i \in \mathcal{E}$ and a delay scenario d we denote by $d(i)$ the delay of event i, that is $d(i) := \pi_i(d) - \pi_i$.

To calculate $d(i)$ directly, we can modify (5.77) in the following way:
First note that for an activity (i, j) we have

$$\begin{aligned} d(i) + d_{ij} \leq wt_{ij} &\Leftrightarrow \pi_i + d(i) + d_{ij} + l_{ij} \leq \pi_i + l_{ij} + wt_{ij} \\ &\Leftrightarrow \pi_i(d) + l_{ij} + d_{ij} \leq \pi_j + wt_{ij} \\ &\Leftrightarrow \pi_i(d) + l_{ij} \leq \pi_j + wt_{ij}. \end{aligned}$$

where the last equivalence holds because

- if $(i,j) \in \mathcal{A}_{change}$ we have $d_{ij} = 0$
- if $(i,j) \notin \mathcal{A}_{change}$ we have $wt_{ij} = \infty$.

Thus (5.77) can be transformed to

$$\begin{aligned} d(j) = \pi_j(d) - \pi_j &= \max\{\pi_j, \max_{\substack{i:(i,j)\in\mathcal{A}\\ \pi_i(d)+l_{ij}\le\pi_j+wt_{ij}}} \{\pi_i(d) + l_{ij} + d_{ij}\}\} - \pi_j \\ &= \max\{0, \max_{\substack{i:(i,j)\in\mathcal{A}\\ \pi_i(d)+l_{ij}\le\pi_j+wt_{ij}}} \{\pi_i + d(i) + l_{ij} + d_{ij} - \pi_j\}\} \\ &= \max\{0, \max_{\substack{i:(i,j)\in\mathcal{A}\\ \pi_i(d)+l_{ij}\le\pi_j+wt_{ij}}} \{d(i) + d_{ij} - s_{ij}\}\} \\ &= \max\{0, \max_{\substack{i:(i,j)\in\mathcal{A}\\ d(i)+d_{ij}\le wt_{ij}}} \{d(i) + d_{ij} - s_{ij}\}\}. \end{aligned} \tag{5.78}$$

Furthermore, by $\mathcal{N}(i)$ we will denote the events and activities of the network $\mathcal{N}$ from which a directed path to i exists in $\mathcal{N}$. We will refer to $\mathcal{N}(i)$ also as the *network preceding i*.

According to the delay propagation specified in (5.77), delays on activities not lying on a directed path toward j^* cannot influence $d(j^*)$. It hence suffices to consider only the preceding network of an event i when calculating the possible delay at this event. This is summarized in the following observation.

Lemma 5.29. *Consider an event $j^* \in \mathcal{E}$ and a delay scenario $d \in \mathcal{U}_\epsilon^K$ with $d(j^*) = D$. Then it holds that $d'(j^*) = d(j^*)$ for the delay scenario d' defined as*

$$d'_a := \begin{cases} d_a, & \text{if } a \in \mathcal{N}(j^*) \\ 0, & \text{if } a \in \mathcal{N} \setminus \mathcal{N}(j^*). \end{cases}$$

Note that if $d \in \mathcal{U}_\epsilon^K$ for a given K, also $d' \in \mathcal{U}_\epsilon^K$. Consequently, when trying to solve (DA) for an event j^* in $\mathcal{N}$, from now on we will restrict to delay scenarios having only delays in $\mathcal{N}(j^*)$.

In particular, in the following lemmata we show that we can assume that all delays lie on one single path toward j^*. This property is crucial for solving (DA). The following lemma is a direct consequence of (5.77).

Lemma 5.30. *If for an event $j^* \in \mathcal{E}$ and a delay scenario $d \in \mathcal{U}_\epsilon^K$ it holds that $d(j^*) > 0$, then there is at least one non-empty directed path P toward j^* such that for every $(i,j) \in P$*

$$\pi_j(d) = \pi_i(d) + l_{ij} + d_{ij} \text{ and} \tag{5.79}$$

$$wt_{ij} \ge \pi_i(d) - \pi_i - s_{ij}. \tag{5.80}$$

P contains at least one source delay.

Proof. According to (5.76),

$$\pi_j(d) \geq \pi_i(d) + l_{ij} + d_{ij}.$$

Using $\pi_j = \pi_i + l_{ij} + s_{ij}$, the delay propagation rule (5.77) guarantees that there always is an $a = (i, j) \in \mathcal{A}$ such that the relations in (5.79) and (5.80) hold. □

We will call such a path P a *critical path for j^* and d*. In the following we will use the expression "a path P is *maintained* for delay scenario d" to express that all connections on P are maintained in delay scenario d.

The observation of the following lemma will often be used in the following lemmata:

Lemma 5.31. *Let $j^* \in \mathcal{E}$ be an event, $d \in \mathcal{U}_\epsilon^K$ a delay scenario and P a critical path for j^* and d. Let d' be a delay scenario obtained by decreasing the source delay of delay scenario d on one activity of P by 1. Then either*

1. *P is still a critical path for d' and j^* and $d'(j^*) = d(j^*) - 1$,*
2. *$d'(j^*) > d(j^*) - 1$, or*
3. *$d'(j^*) < d(j^*) - 1$ and P is not maintained for delay scenario d'. That is, there is a connection (i, j) on P such that $d(i) \leq wt_{ij} + s_{ij}$ and $d'(i) > wt_{ij} + s_{ij}$.*

The next lemma shows that when we have a delay scenario causing a delay of D at an event j^*, we can also produce any amount of delay smaller than D at j^* by reducing the source delays in an appropriate way.

Lemma 5.32. *Let $d \in \mathcal{U}_\epsilon^K$ be a delay scenario and $j^* \in \mathcal{E}$ an event with $d(j^*) = D$ for a $D \in \mathbb{N}^{\geq 1}$. Then there is a delay scenario d' with $d'_a \leq d_a$ for every $a \in \mathcal{A}$ and $d'(j^*) = D - 1$.*

Proof. We define a topological ordering on $\mathcal{N}$ by saying that for two events $i, j \in \mathcal{E}$ $i \prec j$ if there is a directed path from i to j in $\mathcal{N}$. We show Lemma 5.32 inductively along the topological ordering.

First note that for all topologically minimal events j^*, we have that $d(j^*) = 0$. Thus, there is nothing to show for these events.

Now consider an event j^* and suppose that the statement of the lemma has been shown for all its predecessors. Let $D := d(j^*)$ be the delay caused by delay scenario d in j^*. We now describe how to construct a delay scenario d' with $d'(j^*) = D - 1$ and $d'_a < d_a$ for all $a \in \mathcal{A}$. For that, we iteratively apply the following procedure:
Let P be a critical path for delay scenario d and event j^*. Let j' be the predecessor of j^* on P. Then, due to (5.78), $d(j') = D - d_{(j',j^*)} + s_{(j',j^*)}$. Depending on the delay $d_{(j',j^*)}$ on the arc (j', j^*) we distinguish the following cases.

1. If $d_{(j',j^*)} > 0$, let $\hat{d}$ denote the delay scenario defined by $\hat{d}_{(j',j^*)} := d_{(j',j^*)} - 1$ and $\hat{d}_a := d_a$ for all $a \neq (j', j^*)$. Then, if P is still a critical path for j^* and $\hat{d}$, $\hat{d}(j^*) = \hat{d}(j') + \hat{d}_{(j',j^*)} - s_{(j',j^*)} = (D - d_{(j',j^*)} + s_{(j',j^*)}) + (d_{(j',j^*)} - 1) - s_{(j',j^*)} = D-1$ and we set $d' := \hat{d}$. Otherwise we obtain $\hat{d}(j^*) = D$ and repeat the procedure for $d := \hat{d}$.

2. If $d_{(j',j^*)} = 0$, we have $d(j') = D + s_{(j',j^*)}$. Let $\hat{d}$ denote a delay scenario with $\hat{d}_a \leq d_a$ for every $a \in \mathcal{A}$ and $\hat{d}(j') = D + s_{(j',j^*)} - 1$. Such a delay scenario exists because of the induction hypothesis. Then, if P is still a critical path for j^* and $\hat{d}$, $\hat{d}(j^*) = \hat{d}(j') + \hat{d}_{(j',j^*)} - s_{(j',j^*)} = (D - d_{(j',j^*)} + s_{(j',j^*)}) + (d_{(j',j^*)} - 1) - s_{(j',j^*)} = D-1$ and we set $d' := \hat{d}$.
Note that if $(j', j^*) \in \mathcal{A}_{change}$, we have $wt_{(j',j^*)} \geq d(j') - s_{(j',j^*)}$ because otherwise P could not be a critical path for d and j^*. Thus it also holds that $wt_{(j',j^*)} \geq d(j') - s_{(j',j^*)} - 1 = \hat{d}(j') - s_{(j',j^*)}$, which means that (j', j^*) is maintained for delay scenario $\hat{d}$. Hence if P is not a critical path for $\hat{d}$ and j^*, it holds that $\hat{d}(j^*) > D - 1$. In this case we repeat the procedure for $d := \hat{d}$.

In every step of this procedure, the delay in j^* is bigger or equal to $D-1$. Furthermore, the overall delay in the network is reduced. As the total amount of delay in the network is finite, the procedure stops with a delay scenario d' with $d'(j^*) = D - 1$. □

Corollary 5.33. *If there is a delay scenario $d \in \mathcal{U}_\epsilon^K$ such that for an event $j^* \in \mathcal{E}$ it holds that $d(j^*) = D$, then for every natural number $D' \leq D$, there is a delay scenario d' with $d'_a \leq d_a$ for every $a \in \mathcal{A}$ and $d'(j^*) = D'$.*

The following Lemma 5.34 allows us to consider only delay scenarios where all delays lie on a critical path toward the considered event in (DA). In cases where we are interested in the delay of a specific event j^*, we will refer to delay scenarios where all occurring source delays lie on a critical path toward j^* as *path delay scenarios toward j^** or just as a *path delay scenario* whenever the event j^* is implicitly specified. For the sake of brevity, we will say that an event or an activity *lies on a path delay scenario* instead of saying that it is contained in the critical path belonging to a path delay scenario.

Lemma 5.34. *Let j^* be an event in $\mathcal{E}$ and $d \in \mathcal{U}_\epsilon^K$ a delay scenario. Then there is a delay scenario $d' \in \mathcal{U}_\epsilon^K$ with $d'_a \leq d_a$ $\forall a \in \mathcal{A}$ and all activities a with $d'_a > 0$ lying on a critical path P' toward j^* such that $d'(j^*) = d(j^*)$.*

Proof. Let the events be topologically sorted. We show this lemma inductively along the topological ordering. First note that for all topologically minimal events j^* it holds that $d(j^*) = 0$, thus there is nothing to show for these events. Now consider an event j^* and suppose that the statement of the lemma has been shown for all its predecessors. We construct a *path delay scenario* for j^* as follows:

Let j' be the predecessor of j^* on a critical path P' for d and j^*. Let $\hat{d}$ be a delay scenario and $\hat{P}$ be a path ending in j' such that $\hat{d}_a \leq d_a$ for all $a \in \mathcal{A}$, $\hat{d}_a = 0$ for all $a \notin \hat{P}$, and $\hat{d}(j') = d(j')$. Such a delay scenario exists due to the induction hypothesis. For j', $\hat{d}$ and P' we apply the following procedure:
Let d' be the delay scenario with $d'_{(j',j^*)} := d_{(j',j^*)}$ and $d'_a := \hat{d}_a$ for all $a \neq (j', j^*)$. Then P' is a critical path for d' and j'. Note that if $(j', j^*) \in \mathcal{A}_{change}$, it holds that $wt_{(j',j^*)} \geq d(j') + d_{(j',j^*)} - s_{(j',j^*)} = d'(j') + d'_{(j',j^*)} - s_{(j',j^*)}$, thus (j', j^*) is maintained in delay scenario d' and $d'(j^*) \geq d'(j') - d'_{(j',j^*)} + s_{(j',j^*)} = d(j^*)$.

1. If and only if $d'(j^*) = d(j^*)$, the union $P' \cup (j', j^*)$ is a critical path for d' and j^*. In this case, the statement of the lemma is shown.

2. Otherwise, due to Corollary 5.33, there is a delay scenario $\tilde{d}$ such that $\tilde{d}(j^*) = d(j^*)$ and $\tilde{d}_a \leq d'_a$ for all $a \in \mathcal{A}$. Let $\tilde{j}$ be the predecessor of j^* on a critical path $\tilde{P}$ for j^* and $\tilde{d}$. If $\tilde{j} = j'$, $\tilde{d}$ is a path delay scenario for j^* and we are done. Otherwise, we repeat the procedure after renaming $j' := \tilde{j}$, $\hat{d} := \tilde{d}$ and $\hat{d} := \tilde{d}$.

Note that every time Condition 1 is not met, the total amount of source delay is decreased by at least one, therefore the procedure terminates in finite time. □

Considering only path delay scenarios is the basic idea behind the dynamic-programming algorithm. Note that when $d \in \mathcal{U}_\epsilon^K$ for given K and ϵ, also the path delay scenario d' constructed like in Lemma 5.34 is contained in $\mathcal{U}_\epsilon^K$. As a result, every feasible delay scenario can be turned into a feasible path delay scenario causing the same delay in the regarded event. Consequently, in the following for solving the problem (DA) we will only look at path delay scenarios.

Using (5.78) we observe the following:

Lemma 5.35. *Let $(i, j) \in \mathcal{A}_{drive} \cup \mathcal{A}_{wait}$ and $D \in \mathbb{N}$. If there is a path delay scenario $d \in \mathcal{U}_\epsilon^K$ that contains (i, j) and at most k large source delays lie before j, such that the delay in j is $d(j) = D$, then for d it holds that*

- *either $d(i) \geq d(j) - \varepsilon_{ij} + s_{ij}$ and at most k large delay scenarios lie before i, or*
- *$d(i) \geq d(j) - d_{ij}^{max} + s_{ij}$ and at most $k - 1$ large source delays lie before i.*

Let $(i, j) \in \mathcal{A}_{change}$ and $D \in \mathbb{N}$. If there is a path delay scenario d that contains (i, j) and at most k large source delays lie before j, such that the delay in j is $d(j) = D$, then for d it holds that

- *$d(j) - s_{ij} \leq wt_{ij}$, $d(i) \geq d(j) - s_{ij}$ and at most k large source delays lie before i.*

Based on these observations, we can build a polynomial time dynamic-programming algorithm which for a given event j^* and a number D determines whether there is a path delay scenario that causes a delay of D at j^*. Starting with j^*, the algorithm goes backwards in the network and successively sets the event labels $d(j, k)$ which indicate how much delay is needed at event j to cause a delay of D at j^* under the assumption that at most $K-k$ large source delays on activities succeeding j are set. Algorithm 5 summarizes this in pseudo code.

Theorem 5.36. *For a given event $j^* \in \mathcal{E}$, an uncertainty set $\mathcal{U}_\epsilon^K$ and a number $D \in \mathbb{N}$, Algorithm 5 solves the problem (DA) in time $O(|\mathcal{A}|K)$:*

- *If there is a delay scenario $d \in \mathcal{U}_\epsilon^K$ with $d(j^*) = D$, Algorithm 5 returns "Yes".*
- *Otherwise, Algorithm 5 returns "No".*

We first prove the following statement:

Lemma 5.37. *Let $\mathcal{N}$, $\mathcal{U}_\varepsilon^K$, $j^* \in \mathcal{E}$, a predecessor $i \in \mathcal{E}$ of j^*, $D \in \mathbb{N}$, and a natural number $k \leq K$ be given. Let $\tilde{\mathcal{N}}$ denote the network we obtain from $\mathcal{N}$ be adding a node i' and an arc (i', i) with $d^{max}_{(i',i)} = \infty$. Then the number $d(i,k)$, as calculated in Algorithm 5, is the minimal number x such that there is a path P from i to j^* in $\mathcal{N}$ and a delay scenario $d \in \mathcal{U}_\varepsilon^k$ with $d_a = 0$ $\forall a \notin P$ such that for the delay scenario $\tilde{d}$ defined as*

$$\tilde{d}_a := \begin{cases} d_a & \text{if } a \in \mathcal{N} \\ x & \text{if } a = (i', i) \end{cases}$$

on $\tilde{N}$ it holds that $\tilde{d}(j^*) = D$.

Proof of Lemma 5.37. We show inductively that for an event i, once all activities $a' = (i,j) \in \mathcal{N}(j^*)$ have been regarded in steps 4-17, the label $d(i,k)$ has the required property.
In Step 2, $d(j^*,k)$ is correctly set to D for all $k = 1, 2, \ldots, K$.
Let i be a predecessor of j^* in $\mathcal{N}(j^*)$ and denote by $\mathcal{A}(i,j^*)$ all activities starting in i in the network $\mathcal{N}(j^*)$. Suppose that for an event i we have already considered the end events j of all outgoing activities $a' = (i,j) \in \mathcal{A}(i,j^*)$ and that for each such j the induction hypothesis holds. Note that indeed, since the activities in $\mathcal{N}(j^*)$ are ordered topologically, under the induction hypothesis an activity a' will not be considered before all its successors in $\mathcal{N}(j^*)$ are examined and all labels $d(j,k)$ for all $k = 0, \ldots, K$ and all succeeding events j are permanently set. Denote

$$\begin{aligned} \mathcal{A}^{change}(i,j^*) :=& \{a' \in \mathcal{A}(i,j^*) \cap \mathcal{A}_{change} : d(j,k) - s_{ij} \leq wt_{ij}\} \\ \mathcal{A}^{s}(i,j^*) :=& \{a' \in \mathcal{A}(i,j^*) \cap (\mathcal{A}_{drive} \cup \mathcal{A}_{wait})\}. \end{aligned}$$

Note that $\tilde{d}(i) = d(i,k)$, hence according to Lemma 5.35 we have

$$\begin{aligned} d(i,k) = \min\{x : \exists P \text{ path from } i \text{ to } j^*, \exists d \in \mathcal{U}_\varepsilon^k \text{ with } d_a = 0\ \forall a \notin P, \\ \text{such that } \tilde{d}(j^*) = D\} \end{aligned}$$

$$= \min\Big\{ \min_{a'=(i,j)\in\mathcal{A}^s(i,j^*)} \{d(j,k)+s_{ij}-\varepsilon_{ij}\}, \\ \min_{a'=(i,j)\in\mathcal{A}^s(i,j^*)} \{d(j,k+1)+s_{ij}-d_{ij}^{max}\}, \\ \min_{a'=(i,j)\in\mathcal{A}^{change}(i,j^*)} \{d(j,k)+s_{ij}\}\Big\}$$

as calculated by Algorithm 5 in Steps 4-17.

□

Proof of Theorem 5.36. If the algorithm terminates with "No", then for every event $i \in \mathcal{N}(j^*)$ it holds that $d(i,0) > 0$. According to Lemma 5.37, this implies that a delay path cannot start in any event i. Thus there is no path delay scenario in $\mathcal{U}_\varepsilon^K$. Due to Lemma 5.34 we can conclude that there is no delay scenario $d \in \mathcal{U}_\varepsilon^K$ such that $d(j^*) = D$.

If the algorithm terminates with a delay scenario d, there is an event j_0 such that $d(j_0,k^*) \le 0$ for a $k^* \in \{0,\dots,K\}$. Let $P = (j_0,j_1,\dots,j_l)$ with $j_l := j^*$ be the path from j_0 to j^* corresponding to the path delay scenario d^*. Inductively we see that due to the construction of P we have

$$d(j_1,k(j_1)+k^*) = d^*_{j_0j_1} - s_{j_0j_1} \tag{5.81}$$

$$d(j_{i+1},k(j_{i+1})+k^*) = d(j_i,k(j_i)+k^*) + d^*_{j_ij_{i+1}} - s_{j_ij_{i+1}} \quad \text{for } i=1,\dots,l-1, \tag{5.82}$$

where $k(j_i)$ is the number of large source delays before j_i on path P in delay scenario d^*.

We now show inductively, that for all $i = 1,\dots,l$ there is a path delay scenario $d^i \in \mathcal{U}_\varepsilon^K$ with

$$d^i(j_i) = d(j_i,k(j_i)+k^*).$$

This follows directly from (5.81) for $d^1(j_1)$. Now suppose that the induction hypothesis holds for $j_1,\dots,j_{i-1}$. From (5.78) it follows that

$$d^{i-1}(j_i) \ge d^{i-1}(j_{i-1}) + d^{i-1}_{j_ij_{i-1}} - s_{j_ij_{i-1}} = d(j_{i-1},k(j_{i-1})+k^*) + d^*_{j_ij_{i-1}} - s_{j_ij_{i-1}} \tag{5.83}$$

$$= d(j_i,k(j_i)+k^*). \tag{5.84}$$

The statement now follows from Lemma 5.32. Thus there is a delay scenario $d^l \in \mathcal{U}_\varepsilon^K$ such that

$$d^l(j^*) = d(j^*,k(j^*)+k^*) = D.$$

Thus, Algorithm 5 is correct. Since every step inside the for-loops of the algorithm can be executed in linear time, the total running time of the algorithm is in $O(|\mathcal{A}|K)$ with $K \le |\mathcal{E}|$.

□

The set $\mathcal{A}^{acc}$ can now be obtained by using Algorithm 5 for every $a \in \mathcal{A}_{change}$ with $D := s_a + wt_a + 1$. The total complexity to do so is therefore $O(|\mathcal{A}||\mathcal{A}_{change}|K)$.

Algorithm 5 (Delay accumulation)

```
Require: Event-activity network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ with $\mathcal{A}$ topologically sorted (backwards),
    uncertainty set $\mathcal{U}_\epsilon^K$, event $j^*$, number $D$
Ensure: "Yes", if there is a delay scenario $d \in \mathcal{U}_\epsilon^K$ that causes a delay of at least $D$
    in $j^*$. "No" otherwise.
1: Set $d(j,k) = \infty$, $succ(j,k) = \emptyset$ for all $k = 1, \dots, K$, $j \in \mathcal{E}$.
2: Set $d(j^*, K) = D$ for $k = 1, 2, \dots, K$.
3: for $a \in \mathcal{A}$, topologically sorted backwards do
4:     Let $(i,j) = a$.
5:     for $k = K, K-1, \dots, 1$ do
6:         if $a \in \mathcal{A}_{drive} \cup \mathcal{A}_{wait}$ then
7:             if $d(i,k) > \min\{d(j,k) + s_{ij} - \epsilon_{ij}, d(j,k+1) + s_{ij} - d_{ij}^{max}\}$ then
8:                 $d(i,k) = \min\{d(j,k) + s_{ij} - \epsilon_{ij}, d(j,k+1) + s_{ij} - d_{ij}^{max}\}$,
9:                 set $succ(i,k) := (j,k)$ or $succ(i,k) := (j,k+1)$ respectively.
10:            end if
11:        else if $a \in \mathcal{A}_{change}$ and $d(j,k) < wt_{ij}$ and $d(i,k) > d(j,k) + s_{ij}$ then
12:            $d(i,k) = d(j,k) + s_{ij}$ and $succ(i,k) := (j,k)$
13:        end if
14:        if $d(i,k) \leq 0$ then
15:            return "Yes".
16:        end if
17:    end for
18: end for
19: return "No".
```

5.4.5 Lightly Robust Timetable Information

Allowing only strictly robust solutions will often lead to paths with very long travel time that will probably not be accepted by the passengers. A promising alternative is light robustness. In our setting this means that the output for the passenger should be a path with reasonable length, that is, its length should not exceed the length of a nominal optimal path by too much. Among all solutions satisfying this criterion one looks for the "most robust" one, which we define as the one with the fewest number of for the unreliable transfers is given, weights can be introduced to differ between the grade of unreliability for these activities.

For the robust timetable information problem we hence allow that the path gets longer in order to make it more robust: Let $f^* := f(P^*)$ denote the length of a shortest path for a request Req $= (u, v, t_{\text{request}})$ in the undisturbed scenario, and B a parameter bounding the allowed increase in travel time.

(Light-robust-path) Given a network $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ a timetable π, a request Req consisting of an origin u, a destination v and a time t_{request}, and the set of strictly robust changing activities $\mathcal{A}^{\text{SR}}$, find a path P with length smaller or equal to $f^* + B$

that contains as few as possible changing activities not contained in $\mathcal{A}^{\mathrm{SR}}$.

Given the set of strictly robust changing activities $\mathcal{A}^{\mathrm{SR}}$ as defined in Section 5.4.3, we can find such a path using a shortest path algorithm minimizing the number of changing activities classified as being not strictly robust in an event-activity network where we exclude all events that take place later than $f^* + B$. This leads to the following lemma:

Lemma 5.38. *Given the set of strictly robust connections $\mathcal{A}^{\mathrm{SR}}$, (Light-robust-path) can be solved in time $O(|\mathcal{A}| + |\mathcal{E}| \log(|\mathcal{E}|))$.*

Proof. Our solution approach is the following: Let u and v be the origin and destination of the considered passenger request and $t_{request}$ be its earliest starting time. First, reduce the network such that it only contains events $i \in \mathcal{N}(v) = (\mathcal{E}(v), \mathcal{A}(v))$ with $\pi_i \leq t_{request} + f^* + B$. Now the reduced network contains only paths from u to v that reach the destination within the required time limit. Within this network we look for a path with a minimal number of activities in $\mathcal{A}_{change} \setminus \mathcal{A}^{\mathrm{SR}}$. This can be done in time $O(|\mathcal{A}(v)| + |\mathcal{E}(v)| \log(|\mathcal{E}(v)|)) = O(|\mathcal{A}| + |\mathcal{E}| \log(|\mathcal{E}|))$ using a shortest path algorithm by weighting the activities in $\mathcal{A}_{change} \setminus \mathcal{A}^{\mathrm{SR}}$ with e.g. 1, and use a cost of zero for all other activities. □

Note that we assumed in the problem formulation of (Light-robust-path) that the set of strictly robust activities $\mathcal{A}^{\mathrm{SR}}$ is given. As we have seen in Theorem 5.28, determining the set $\mathcal{A}^{\mathrm{SR}}$ is strongly NP-hard in general. For finding a heuristic solution we can again consider the subset $\mathcal{A}^{acc}$ instead of $\mathcal{A}^{\mathrm{SR}}$.

Compared to the approach of strictly robust timetable information, lightly robust paths are not necessarily maintained under disruptions. However, since passengers may be willing to sacrifice some robustness for shorter planned travel times, this trade-off may be beneficial.

5.4.6 Empirical Evaluation

Environment. All experiments were run on a PC (Intel(R) Xeon(R), 2.93GHz, 4MB cache, 47GB main memory under Ubuntu Linux version 10.10). Only one core has been used by our program. Our code is written in C++ and has been compiled with g++ 4.4.3 and compile option -O3.

Test instances and delay scenarios. Our computational study is based on the German train schedule of 2011, from which we derived two event-activity networks, namely a large-scale network with all trains (referred to as *complete network*) and a smaller network restricted to high-speed trains of the train categories Intercity Express (ICE), Intercity (IC), and Eurocity (EC), called *high-speed network* in the sequel. Table 5.18 states the main characteristics of both networks.

We generated transfer activities between pairs of trains at the same station, if the departing train is scheduled to depart not later than 120 minutes after the arrival time

characteristic	high-speed network	complete network
# trains	870	38985
# events	37566	2041594
# stations	538	6866
# transfer activities	58610	6097308
aver. nominal travel time	433 min	410 min
aver. # transfers per query	1.75	1.88

Table 5.18 Characteristics of the two networks, given numbers correspond to two successive days, namely March 10 and 11, 2011, and characteristics of test queries.

of the feeding train. Note that this gives us an implicit bound of 120 minutes for the maximum delay that robust paths can compensate for. However, we believe that this is sufficient for any reasonable strategy of robust pre-trip timetable information in practice. We applied the following waiting rules: High speed trains wait for each other at most 3 minutes. Trains of other train categories do not wait for each other.

Passenger path requests have been generated by randomly chosen origins and destinations. Start times are chosen randomly in the interval of the first 12 hours of the day. To avoid trivial requests, we included only those requests for which the distance between start and destination is at least $150km$ and which require in the nominal scenario at least one transfer. The average travel time is 433 min and 410 min for the high-speed and complete network, respectively. The average number of transfers required in the optimal path in the nominal scenario is 1.75 and 1.88, again for the high-speed and complete network, respectively.

In our experiments, we consider the scenario set $\mathcal{U}_\epsilon^K$ as defined in (5.75). Our artificial delay scenarios are characterized by three parameters, ε, A, and K:

- The parameter ε controls the maximal size of "small delays" which can occur in our model on every activity. This parameter is from the set $\{0.01l_a, 0.02l_a, \ldots, 0.1l_a\}$, i.e., small delays are chosen as a fraction of the nominal length l_a of waiting and driving activities.
- The second parameter A specifies the maximal size of a "large delay" if it occurs. Here we add the constant A to the maximal small delay of the activity. In our experiments, we used $A \in \{5, 10, 15, 20\}$ (in minutes).
- Finally, our third parameter K specifies the maximum number of "large delays" which may occur on a path. We assume that a passenger will be affected only by a small number of such "large delays", therefore we have K varied among $\{0, 1, 2, 3\}$.

Each parameter set can be interpreted as defining a certain "level of guaranteed reliability": Strict robust timetable information will deliver only paths for which all changing activities are immune against all delay scenarios described by this parameter set. Hence, the larger we choose these parameters, the stronger guarantees we obtain.

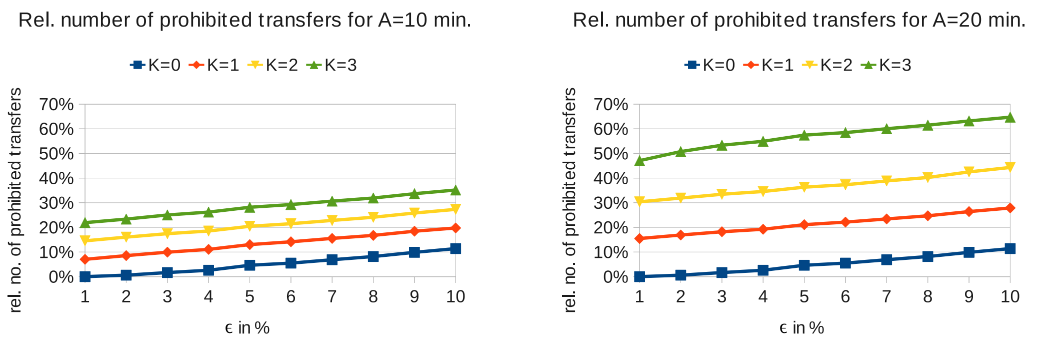


Figure 5.18 High-speed network: The fraction of transfer activities which are infeasible according to delay accumulation for different parameter sets of the delay scenarios.

Experiment 1 — strictly robust transfer activities. In our first experiment, we want to study how many transfer activities which exist in the nominal scenario are not strictly robust? And how does this number depend on the parameters of the delay scenario? To determine strictly robust transfer activities, we use our conservative over-approximation Algorithm 5 to compute the set $\mathcal{A}^{acc}$. The CPU time to compute $\mathcal{A}^{acc}$ is about 2 minutes per parameter set for the high-speed network, while the complete German train network requires about 124 hours per parameter set. Due to the excessive running times, we used only a restricted set of parameter combinations for the complete train network. However, for both types of networks we observe that a considerable fraction becomes infeasible with increasing size of the delay parameters, see Figures 5.18 and 5.19. In timetabling, one usually tries to optimize transfer times for transfer activities. This leads for both network types to a certain base of about 5% of transfer activities which are not strictly robust already if the “large delay” parameter is set to $A = 5$. But Figure 5.19 also reveals that the complete network has a larger fraction of transfer activities which become infeasible for larger delays (increased parameter A). In the high-speed network, transfer times are either planned (and therefore short) or they are relatively long (due to its partially periodic structure with a period length of one or two hours), while this distribution is more balanced in the complete network. In this sense, the high-speed network contains a larger “natural” slack.

Futhermore, more transfers are kept in the high-speed network due to the waiting time rule:High-speed trains wait up to three minutes for each other while trains of other categories do not wait for each other, which is why small delays are more easily absorbed than in the complete network.

Experiment 2 — price of strict robustness. With this experiment we want to study quantitatively by how much the planned travel time increases when we compare strictly robust paths with nominal optimal paths. To this end, we have built 1000 random requests (the same set for each parameter setting; in our evaluations we always average

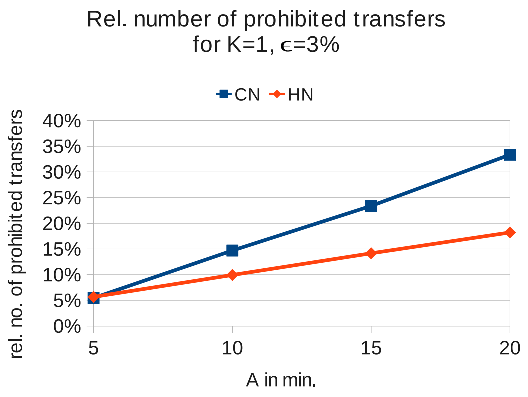


Figure 5.19 Network comparison: Complete network (CN) vs. high-speed network (HN). The fraction of transfer activities which are infeasible according to delay accumulation for different parameter sets of the delay scenarios.

over these requests). As a basis for our comparison, for each request we determine the earliest arrival time with respect to the planned schedule (nominal scenario). Among all paths with earliest arrival time we determine the minimum number of transfers. To solve these requests, we use a (standard) multi-criteria, time-dependent shortest path algorithm. Our implementation reuses the approach described in [BGM10]. For the strictly robust requests the code has been extended to handle “forbidden transfers”. More precisely, it is now possible to specify a list of forbidden transfers between pairs of trains, as computed in Experiment 1 by delay accumulation.

Figure 5.20 shows the average relative increase in travel time induced by the strictly robust paths found using our heuristic estimation of robust changing activities in comparison with optimal paths in the nominal scenario for the high-speed network. The average travel time for the nominal paths is 433 minutes. This implies that the absolute average increase of the travel time in minutes becomes quite large — even for moderate parameter sets. As expected, Figure 5.20 clearly shows that the price of robustness increases monotonically for increasing levels of guaranteed reliability; it grows roughly linearly with respect to parameter ε. Figure 5.21 compares the price of robustness for the two types of networks. We observe that the increase in travel time is significantly higher for the high-speed network than for the complete network. A considerable part of the higher travel times goes hand in hand with a larger minimum slack time.

While these average increases are quite high, a closer look into the distribution of the data reveals an interesting picture; see Figure 5.22 which shows a cumulative diagram for the percentage of passengers with $\leq x$ minutes of travel time increase. Here we use the parameter set $K = 1$, $A = 20$, and $\varepsilon = 0.03$, that is, we allow one large delay of 20 minutes, and arbitrarily many small delays of 3% of the scheduled time for the corresponding activity. The cumulative diagrams show an interesting pattern, with clearly pronounced steps for a travel time increase of 60 and 120 minutes in the case of the high-speed network, and an additional step at 180 minutes for the complete

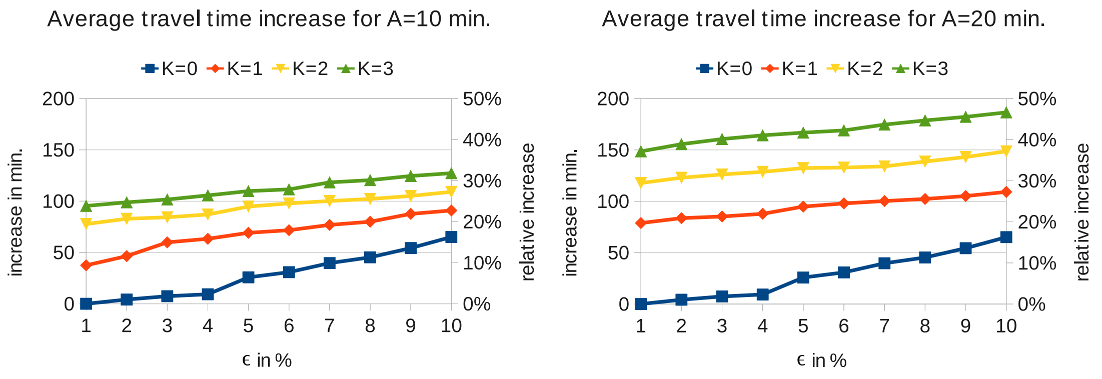


Figure 5.20 High-speed network: The average absolute and relative increase of travel time (in minutes and in %, respectively) for quickest strictly robust paths over optimal paths in the nominal scenario for different parameter sets of the delay scenarios.

network. These steps correspond to the period of 60 and 120 minutes by which many train lines operate. Higher multiples of 60 minutes occur in the complete network for queries with several transfers. For the high-speed network, already about 48% of the test queries have no increase in travel time at all, while only 30% of the test queries for the complete network have this property.

Experiment 3 — price of light robustness. Reusing the set of random requests from Experiment 2, we analyze the price of light robustness based on the subset of robust changing activities calculated using Algorithm 5. The maximum increase of travel time over the nominally smallest travel time was bounded from above by the parameter B (in minutes), with $B \in \{60, 120\}$. The added value of a lightly robust solution in comparison with an optimal solution in the nominal scenario can be measured in two ways:

1. How often is the solution of the lightly robust optimization problem even a strictly robust one?
2. What is the effect on the minimum slack time for changing activities? This number tells us for each passenger the minimum buffer time available for her transfers.

Our detailed analysis starts with the high-speed network (HN). Figure 5.23 (upper left) shows the number of strictly robust paths among our 1000 test queries obtained for the standard non-robust earliest arrival time algorithm (non-robust query), and the corresponding numbers for the lightly robust counterparts with $B = 60$ and $B = 120$. We observe that lightly robust optimization succeeds in many more cases than the non-robust query in finding a strictly robust path within the given “extra time budget” B

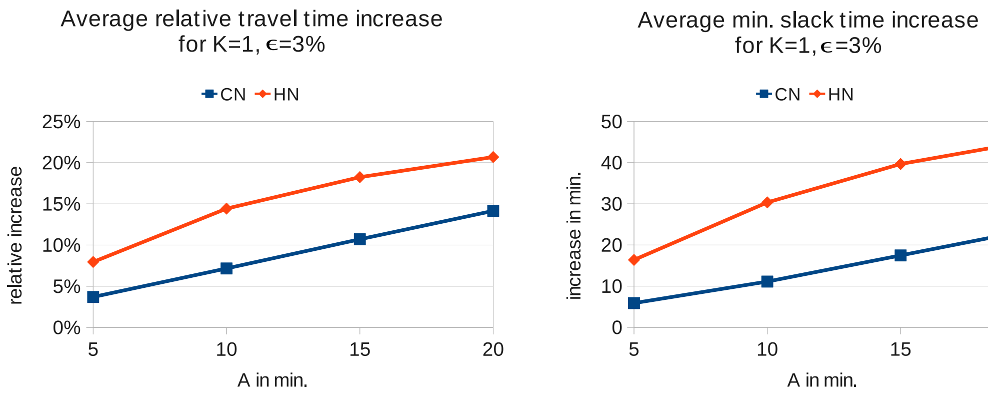


Figure 5.21 Network comparison: Complete network (CN) vs. high-speed network (HN). The relative increase of travel time in % (left) and the average minimum slack time increase in minutes (right), for quickest strictly robust paths over optimal paths in the nominal scenario for different parameter sets of the delay scenarios.

for the travel time increase. We also evaluated by how much the minimum slack time for changing activities increases (upper right part of Figure 5.23) for lightly robust paths in comparison with the nominal case. This measure clearly shows the added reliability achievable by light robustness. The price to achieve this is a relatively moderate average increase of travel time — much more acceptable than for strict robustness (see lower part of the figure).

Figures 5.24 and 5.25 compare the two railway networks with respect to average slack time increase, travel time increase, and the number of cases where the lightly robust solution turns out to use only transfer activities that have been recognized as strictly robust. The two measures for the added robustness are quite similar for both networks, but the price to be paid in terms of increase of travel time seems to be slightly higher for the complete network than for the high-speed network. This is due to the higher train frequencies, which allow passengers to choose from a larger set of possible paths, and therefore allow a better exploitation of the given additional travel time budget of 60 and 120 minutes, respectively.

Note that the characteristics of the query instances are similar for both networks. Hence, the observation that more test queries yield strictly robust paths for ordinary non-robust queries in the high-speed network than in the complete network is a network property, which gives further support to the ”natural slack” explanation, as given in Experiment 1.

5.4.7 Conclusion

Two concepts for calculating robust passenger paths in public transportation networks are proposed: One that searches for routes that will *never fail* for a given set of delay scenarios, and one that finds the most reliable route within a given extra time. Both

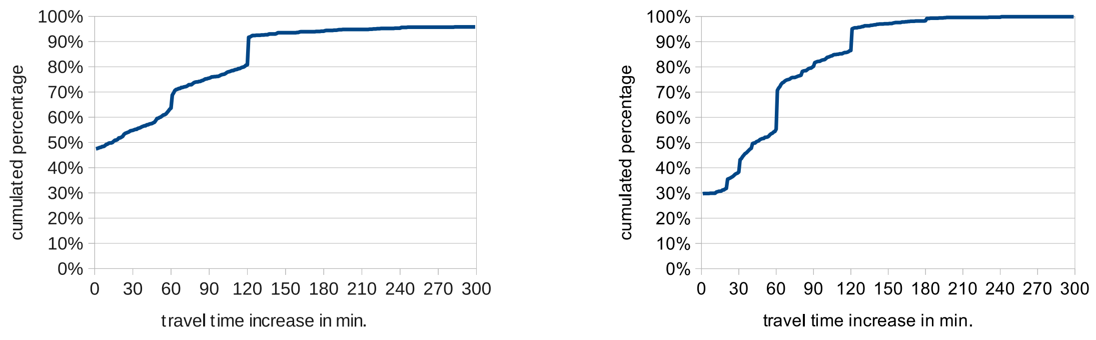


Figure 5.22 High-speed network (left) vs. complete network (right): The cumulative percentage of passengers with less than x minutes of travel time increase for the parameter set $K = 1$, $A = 20$, and $\varepsilon = 0.03$.

problems can be solved efficiently when the set of strictly robust changing activities $\mathcal{A}^{\mathrm{SR}}$ is known. However, determining this set is strongly NP-complete. We propose a dynamic-programming algorithm to find an approximation of this set. In an experimental study, we quantitatively evaluated both robustness concepts using the approximate set of robust transfers. The trade-off between the wish to have more robust paths and the resulting travel time is shown for different levels of protection against delays.

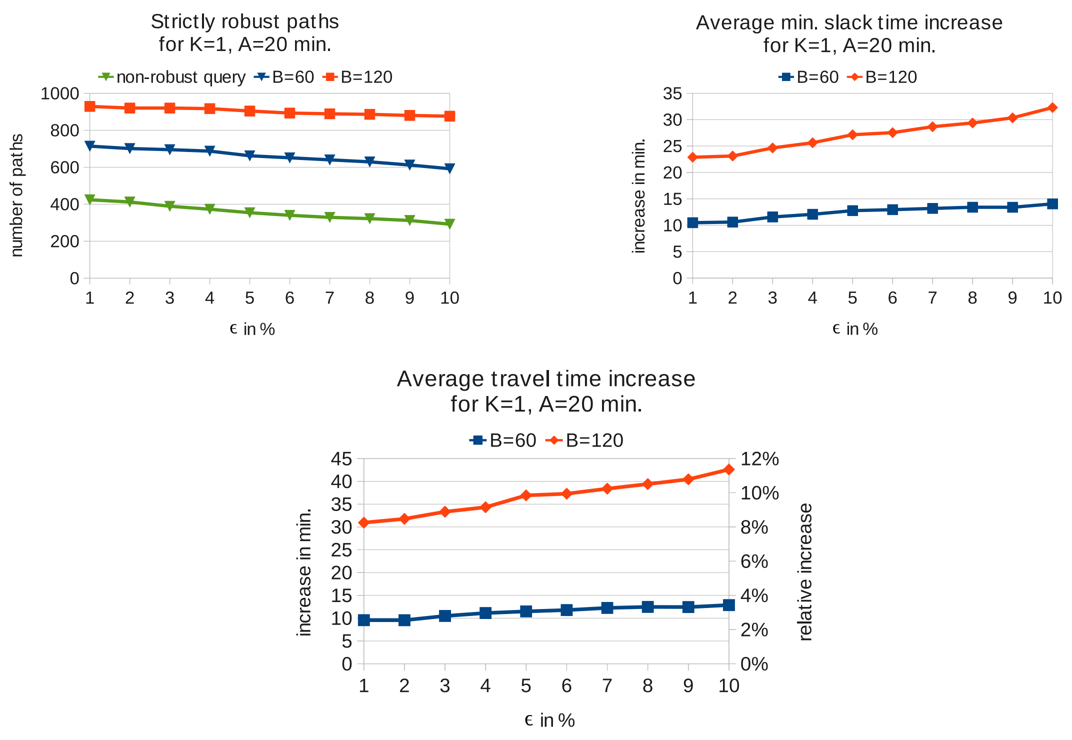


Figure 5.23 Light robustness (high-speed network): The number of strictly robust paths among 1000 test queries obtained for the standard non-robust earliest arrival time algorithm (non-robust query) and the lightly robust counterparts with $B = 60$ and $B = 120$ (upper left). The average increase of minimum slack time on the chosen lightly robust path in comparison with the nominal scenario (upper right), the average increase of travel time in minutes and in % (lower part).

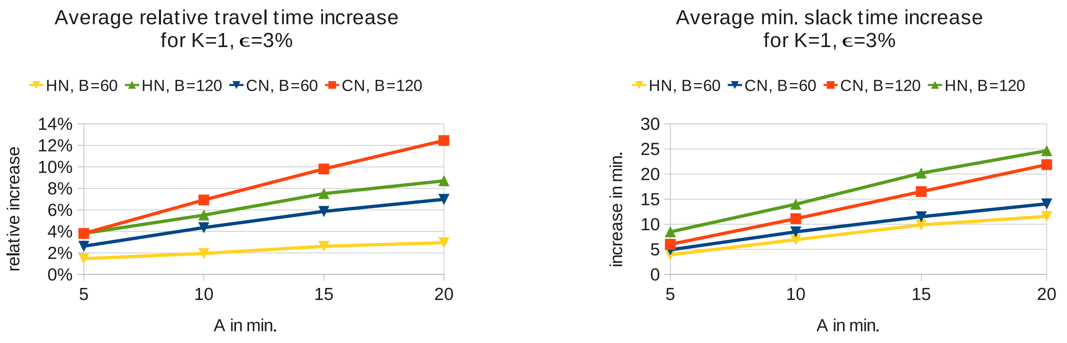


Figure 5.24 Light robustness: The average percentage increase of travel time in minutes (left), and the average increase of minimum slack time on the chosen lightly robust path in comparison with the nominal scenario (right) for different parameter sets of delay scenarios. High-speed network (HN) vs. complete network (CN).

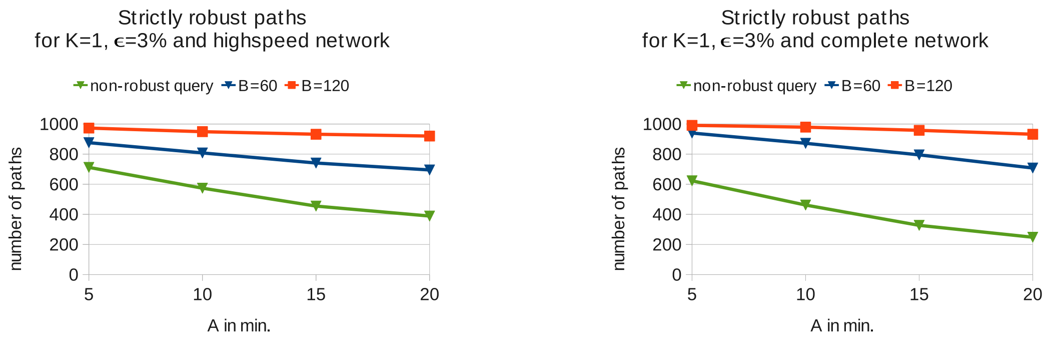


Figure 5.25 Light robustness: The number of strictly robust paths among 1000 test queries obtained for the standard non-robust earliest arrival time algorithm (non-robust query) and the lightly robust counterparts with $B = 60$ and $B = 120$. High-speed network (left) vs. complete network (right).

6 ROPI: A Robust Optimization Programming Interface

Robust optimization aims at pushing optimization techniques to practical applicability. Thus, it is crucial to collect algorithms for robust optimization, and to make them publicly available while being easy-to-use for the practitioner without working deeply into the theoretical background. With ROPI we try to commit to this process. This chapter outlines the basic properties and functionalities, and discusses the differences to other available libraries.

6.1 Introduction

ROPI [Goe12b] is a freely available C++ library that facilitates the applicability of robust optimization models. Following the algorithm engineering paradigm as described in Section 1.3, it implements most of the approaches to robust optimization that are presented in this work.

ROPI can be downloaded from

http://num.math.uni-goettingen.de/˜m.goerigk/ropi

In the following, we describe the status quo of robust optimization libraries in Section 6.2, the distinct library features of ROPI in Section 6.3, and an example application in Section 6.4. This chapter is closed with a discussion of further extensions in Section 6.5.

6.2 Current Software for Robust Optimization

AIMMS for Robust Optimization. AIMMS [Par12], which stands for "Advanced Interactive Multidimensional Modeling System", is a proprietary software that consists of an algebraic modeling language (AML) for optimization problems, and an integrated development environment (IDE) that also allows creating graphical user interfaces. It therefore stands in close competition to other AMLs, such as AMPL or GAMS. In 2011, Paragon Decision Technology, the developer of AIMMS, was part of the winning team of the renowned Franz Edelman Award for Achievement in Operations Research and the Management Sciences. AIMMS supports most well-known solvers, including Cplex, Xpress and Gurobi.

Since 2010, AIMMS offers a robust optimization add-on, which was developed in a partnership with Aharon Ben-Tal. The extension only considers the concepts of strict and adjustable robustness as introduced in Sections 2.1 and 2.3. Using the methodology developed for these concepts, uncertainty sets can be given as the cross-product of intervals, the convex hull of finitely many scenarios, or as an ellipsoid. While mixed-integer programs under the first two kinds of uncertainty can be reformulated to mixed-integer programs again, they become second-order cone programs in the latter case. The respective transformation is automatically done when the model is translated from the algebraic modeling language to the solver.

ROME. While AIMMS focuses on the work of Ben-Tal and co-workers, ROME [GS11c] ("Robust Optimization Made Easy") takes its origins in the work of Bertsimas, Sim and co-workers. ROME is built in the MATLAB environment, which makes it on the one hand intuitive to use for MATLAB-users, but on the other hand lacks the versatility of an AML. As a research project, ROME is free to use. It currently supports Cplex, Mosel and SDPT3 as solver engines.

ROME considers polytopic and ellipsoidal uncertainty sets, that can be further specified using the mean support, the covariance matrix, or directional deviations. It then transforms the uncertain optimization problem to the affinely adjustable robust counterpart; this naturally includes the strictly robust counterpart as a special case.

YALMIP. Similar to ROME, YALMIP [Löf12] is a layer between MATLAB and solver that allows the modeling of optimization problems under uncertainty. Nearly all well-known solvers are supported, including Cplex, Gurobi and Xpress.

The robustness approach of YALMIP is via so-called *filters*: When presented a model with uncertainty, the software checks if one of these filters applies to generate the robust counterpart. Currently, five of these automatic transformations are implemented. A duality filter (which adds dual variables), an enumeration filter (which repeats constraints), an explicit maximization filter (where a worst-case scenario is used), the Pólya filter (which samples scenarios), and an elimination filter (which sets variables affected by uncertainty to 0 and is used as a last resort).

ROPI: A comparison. While each of the aforementioned has their own advantages, ROPI approaches uncertain optimization from a different angle:

C++ interface. The usage of both YALMIP and ROME is restricted to uncertain optimization problems that arise in MATLAB applications. For many academic and industrial applications, this is not sufficient. AIMMS on the other hand grants more flexibility due to its general modeling language, but still cannot compete with direct solver advanced programming interfaces (APIs) in terms of control and speed. ROPI grants the first C++ API to communicate with solvers more directly.

Variety of robustness concepts. While YALMIP, ROME and AIMMS are biased towards the "classic school" of robust optimization, i.e., the work of Ben-Tal, Nemirovski, Bertsimas, Sim, and co-workers, ROPI aims at providing a large choice of possible robust counterparts for an uncertain problem. The user is able to compare the results of various concepts, and choose the solution that best fits his needs.

We summarize some of the differences between the currently available robust optimization software in Table 6.1.

	AIMMS	ROME	YALMIP	ROPI
Interface	AML	MATLAB	MATLAB	C++
License	proprietary	free	free	free
Robustness concepts	strict adjustable	strict adjustable	strict	strict light RecFeas RecOpt
Uncertainty	polyhedral ellipsoidal	polyhedral ellipsoidal	polyhedral conic norm ball	finite (polyhedral)

Table 6.1 Software comparison.

6.3 Library Features

6.3.1 Structure

ROPI is a C++ library providing two main features:

- A user-friendly MIP wrapper class that allows automatic transformation to solver-specific MIP classes. Using this feature, MIPs can be generically written, and solved with whatever solver is currently available. ROPI currently supports transformation to Cplex-, Gurobi-, and Xpress-MIPs. Support for further solvers can be easily added.

- The automatic transformation from a given uncertain MIP to a robust counterpart. Currently implemented are the concepts of strict robustness, light robustness, RecFeas and RecOpt.

In Figure 6.1, the general structure of ROPI is presented.
Visible to the library user are three classes: “Solver”, “LP”, and “Robustness”. Each hides its implementation and specified type within hidden classes, using the *opaque pointer* technique (also known as *pointer to implementation idiom*). This creates the

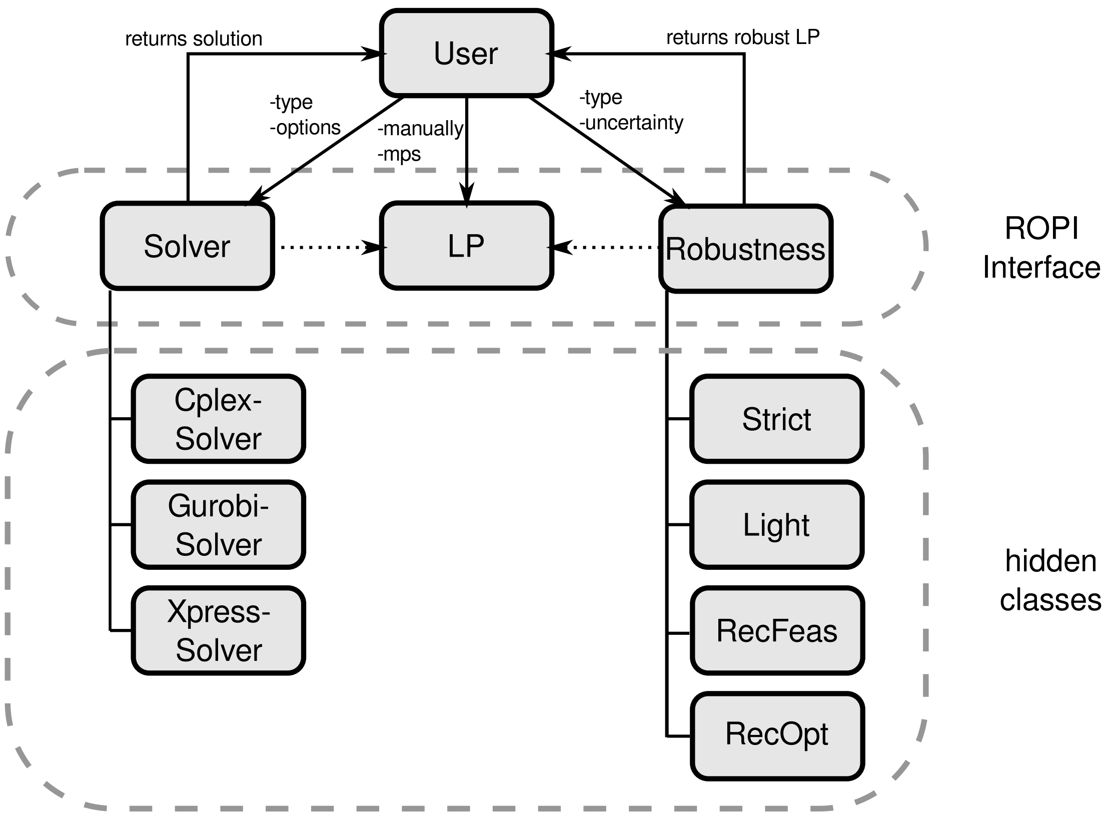


Figure 6.1 Structure of ROPI.

possibility for the user to specify which type of solver or robustness he would like to use during runtime.

The currently implemented type of uncertainty is a finite set of scenarios $\mathcal{U}^f = \{\xi^1, \dots, \xi^N\}$, with further types coming in the next library version. Note that for strict and light robustness, $\mathcal{U}^f$ can also be interpreted as a polytopic uncertainty set with extreme points ξ^i. However, this reduction result does not hold for RecOpt or RecFeas in general, as shown in Section 3.1.4.

For RecOpt and RecFeas, types of distance measures are the l_1 distance $\sum_{i=1}^n |x_i - y_i|$, and the l_∞ distance $\max_{i=1}^n |x_i - y_i|$, while the recovery objective can either be the median or the center. As the philosophy of ROPI is to return a MIP robust counterpart for a given mixed-integer uncertain problem, we can only handle these four cases. However, further algorithmic extensions can be used to handle other distance measures.

6.3.2 Robustness Transformations

We now describe in detail how the robust counterpart to a problem is constructed. We assume an uncertain optimization problem of the form

$$
\begin{array}{lll}
P(A,b) & \min c^t x & \\
& \text{s.t.} & A_1 x \le b_1 \\
& & A_2 x = b_2 \\
& & A_3 x \ge b_3 \\
& & l \le x \le u \\
& & x \in \mathcal{X}
\end{array}
$$

with

$$
A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}
$$

to be given, with a nominal scenario (A^0, b^0). Let the uncertainty set be $\mathcal{U}^f = \{(A^1, b^1), \dots, (A^N, b^N)\} \subseteq \mathbb{R}^{m\times n} \times \mathbb{R}^m$. We show the different robust problem counterparts ROPI can produce. The original MIP size is n variables, and m constraints (plus variable bounds). We denote by $Feas(x, A, b)$ the constraints $A_1 x \le b_1$, $A_2 x = b_2$, and $A_3 x \ge b_3$.

Strict robustness. As described in Section 2.1, the uncertain optimization problem $(P(A,b), (A,b) \in \mathcal{U}^f)$ is transformed to

$$
\begin{array}{ll}
& \min c^t x \\
\text{s.t.} & Feas(x, A^j, b^j) \ \forall j = 0, \dots, N \\
& l \le x \le u
\end{array}
$$

$$x \in \mathcal{X}.$$

The resulting MIP still consists of n variables, but the number of constraints increases to $N(m+1)$.

Light robustness. In a preprocessing step, solve $P(\tilde{A}, \tilde{b})$, and let $c^t x^*$ be the resulting optimal objective value. The lightly robust counterpart as introduced in Section 2.4 for the uncertain optimization problem $(P(\xi), \xi \in \mathcal{U}^f)$ is then given by

$$\begin{aligned}
\min\ & \sum_{i=1}^{m} \gamma_i \\
\text{s.t.}\ & c^t x \leq (1+\rho) c^t x^* \\
& Feas(x, A^0, b^0) \\
& A_1^i x \leq b_1^i + \gamma^1 \ \forall i = 1, \ldots, N \\
& A_2^i x \leq b_2^i + \gamma^2 \ \forall i = 1, \ldots, N \\
& A_2^i x \geq b_2^i - \gamma^2 \ \forall i = 1, \ldots, N \\
& A_3^i x \geq b_3^i - \gamma^3 \ \forall i = 1, \ldots, N \\
& l \leq x \leq u \\
& x \in \mathcal{X},
\end{aligned}$$

where $\gamma = (\gamma_1^t, \gamma_2^t, \gamma_3^t)^t$. The presolving step thus consists of solving a MIP with n variables and m constraints, and the resulting robust counterpart has $n+m$ variables, and $(N+1)m+k$ constraints, where k is the number of equality constraints in the uncertain optimization problem.

RecOpt, simple version. Solve $P(A^i, b^i)$ for all $i = 0, \ldots, N$. Let x^i be an optimal solution to problem $P(A^i, b^i)$. Depending on the recovery distance and the recovery objective, return the problem

	l_1	l_∞
median	$\begin{aligned} \min\ & \sum_{j=1}^{N} \sum_{i=1}^{n} y_{ij} \\ \text{s.t.}\ & -y_{ij} \leq x_i - x_i^j \leq y_{ij} \\ & \forall i = 1, \ldots, n,\ j = 0, \ldots, N \\ & Feas(x, A^0, b^0) \\ & l \leq x \leq u \\ & x \in \mathcal{X}, y \in \mathbb{R}_{\geq 0}^{n \times N} \end{aligned}$	$\begin{aligned} \min\ & \sum_{j=1}^{N} y_j \\ \text{s.t.}\ & -y_j \leq x_i - x_i^j \leq y_j \\ & \forall i = 1, \ldots, n,\ j = 0, \ldots, N \\ & Feas(x, A^0, b^0) \\ & l \leq x \leq u \\ & x \in \mathcal{X}, y \in \mathbb{R}_{\geq 0}^{N} \end{aligned}$

	l_1	l_∞
center	$\begin{array}{ll} \min & z \\ \text{s.t.} & z \geq \sum_{i=1}^{n} y_{ij} \\ & \forall j = 1, \ldots, N \\ & -y_{ij} \leq x_i - x_i^j \leq y_{ij} \\ & \forall i = 1, \ldots, n,\ j = 0, \ldots, N \\ & Feas(x, A^0, b^0) \\ & l \leq x \leq u \\ & x \in \mathcal{X}, y \in \mathbb{R}^{n \times N}_{\geq 0}, z \in \mathbb{R} \end{array}$	$\begin{array}{ll} \min & z \\ \text{s.t.} & z \geq y_j \\ & \forall j = 1, \ldots, N \\ & -y_j \leq x_i - x_i^j \leq y_j \\ & \forall i = 1, \ldots, n,\ j = 0, \ldots, N \\ & Feas(x, A^0, b^0) \\ & l \leq x \leq u \\ & x \in \mathcal{X}, y \in \mathbb{R}^{N}_{\geq 0}, z \in \mathbb{R} \end{array}$

The presolving step thus consists of $(N+1)$ problems with n variables and m constraints each. Concerning the resulting MIP size, we have:

	#variables	#constraints
median, l_1	$n(N+1)$	$nN+m$
median, l_∞	$n+N$	$nN+m$
center, l_1	$n(N+1)+1$	$N(n+1)+m$
center, l_∞	$n+N+1$	$N(n+1)+m$

The constraints $Feas(x, A^0, b^0)$ ensuring feasibility in the nominal scenario can be left out if not desired, so does the constraint $x \in \mathcal{X}$, and the variable bounds $l \leq x \leq u$, especially for combinable problems.

RecOpt, extended version. In the extended version, we do not precompute the optimal solutions to each of the scenarios, as they might be ambiguous. Instead, we only use the respective optimal objective value to find a recovery robust solution that can be recovered to *any* optimal solution for every scenario.

Solve $P(A^i, b^i)$ for all $i = 0, \ldots, N$, where $i = 0$ denotes again the nominal problem, and let $f^*(A^i, b^i)$ be the optimal objective value of problem $P(A^i, b^i)$. Depending on the recovery distance and objective, return the problem

	l_1	l_∞
median	$\begin{array}{ll} \min & \sum_{j=1}^{N}\sum_{i=1}^{n} y_{ij} \\ \text{s.t.} & -y_{ij} \le x_i - x_i^j \le y_{ij} \\ & \forall i=1,\dots,n,\ j=0,\dots,N \\ & Feas(x,A^0,b^0) \\ & Feas(x^j,A^j,b^j)\ \forall j=0,\dots,N \\ & c^t x^j = f^*(A^j,b^j)\ \forall j=0,\dots,N \\ & x\in\mathcal{X}, y\in\mathbb{R}^{n\times N} \\ & x^j\in\mathcal{X}\ \forall j=0,\dots,N \end{array}$	$\begin{array}{ll} \min & \sum_{j=1}^{N} y_j \\ \text{s.t.} & -y_j \le x_i - x_i^j \le y_j \\ & \forall i=1,\dots,n,\ j=1,\dots,N \\ & Feas(x,A^0,b^0) \\ & Feas(x^j,A^j,b^j)\ \forall j=0,\dots,N \\ & c^t x^j = f^*(A^j,b^j)\ \forall j=0,\dots,N \\ & x\in\mathcal{X}, y\in\mathbb{R}^{N} \\ & x^j\in\mathcal{X}\ \forall j=1,\dots,N \end{array}$

	l_1	l_∞
center	$\begin{array}{ll} \min & z \\ \text{s.t.} & z \ge \sum_{i=1}^{n} y_{ij} \\ & \forall j=1,\dots,N \\ & -y_{ij} \le x_i - x_i^j \le y_{ij} \\ & \forall i=1,\dots,n,\ j=1,\dots,N \\ & Feas(x,A^0,b^0) \\ & Feas(x^j,A^j,b^j)\ \forall j=0,\dots,N \\ & c^t x^j = f^*(A^j,b^j)\ \forall j=0,\dots,N \\ & x\in\mathcal{X}, y\in\mathbb{R}^{n\times N}, z\in\mathbb{R} \\ & x^j\in\mathcal{X}\ \forall j=1,\dots,N \end{array}$	$\begin{array}{ll} \min & z \\ \text{s.t.} & z \ge y_j \\ & \forall j=1,\dots,N \\ & -y_j \le x_i - x_i^j \le y_j \\ & \forall i=1,\dots,n,\ j=1,\dots,N \\ & Feas(x,A^0,b^0) \\ & Feas(x^j,A^j,b^j)\ \forall j=0,\dots,N \\ & c^t x^j = f^*(A^j,b^j)\ \forall j=0,\dots,N \\ & x\in\mathcal{X}, y\in\mathbb{R}^{N}, z\in\mathbb{R} \\ & x^j\in\mathcal{X}\ \forall j=1,\dots,N \end{array}$

The presolving consists of N problems with n variables and m constraints each. For the robust counterpart, depending on whether we use the median or center, and on which l, we have the following numbers of constraints and variables:

	#variables	#constraints
median, l_1	$n(2N+1)$	$N(n+m+1)+m$
median, l_∞	$n+N+nN$	$N(n+m+1)+m$
center, l_1	$n(2N+1)+1$	$N(n+m+2)+m$
center, l_∞	$n+N+nN+1$	$N(n+m+2)+m$

RecFeas. For the RecFeas counterpart, we need to find a feasible solution in every scenario, and a robust solution feasible in the nominal scenario, such that the recovery distance is minimized. We get the following robust counterpart formulations:

	l_1	l_∞
median	$\begin{array}{ll} \min & \sum_{j=1}^{N}\sum_{i=1}^{n} y_{ij} \\ \text{s.t.} & -y_{ij} \leq x_i - x_i^j \leq y_{ij} \\ & \forall i = 1,\ldots,n,\ j = 1,\ldots,N \\ & Feas(x^j, A^j, b^j)\ \forall j = 0,\ldots,N \\ & \forall j = 1,\ldots,N \\ & x \in \mathcal{X}, y \in \mathbb{R}^{n\times N} \\ & x^j \in \mathcal{X}\ \forall j = 1,\ldots,N \end{array}$	$\begin{array}{ll} \min & \sum_{j=1}^{N} y_j \\ \text{s.t.} & -y_j \leq x_i - x_i^j \leq y_j \\ & \forall i = 1,\ldots,n,\ j = 1,\ldots,N \\ & Feas(x, A^0, b^0) \\ & Feas(x^j, A^j, b^j)\ \forall j = 0,\ldots,N \\ & x \in \mathcal{X}, y \in \mathbb{R}^{N} \\ & x^j \in \mathcal{X}\ \forall j = 1,\ldots,N \end{array}$

	l_1	l_∞
center	$\begin{array}{ll} \min & z \\ \text{s.t.} & z \geq \sum_{i=1}^{n} y_{ij} \\ & \forall j = 1,\ldots,N \\ & -y_{ij} \leq x_i - x_i^j \leq y_{ij} \\ & \forall i = 1,\ldots,n,\ j = 1,\ldots,N \\ & Feas(x, A^0, b^0) \\ & Feas(x^j, A^j, b^j)\ \forall j = 0,\ldots,N \\ & x \in \mathcal{X}, y \in \mathbb{R}^{n\times N}, z \in \mathbb{R} \\ & x^j \in \mathcal{X}\ \forall j = 1,\ldots,N \end{array}$	$\begin{array}{ll} \min & z \\ \text{s.t.} & z \geq y_j \\ & \forall j = 1,\ldots,N \\ & -y_j \leq x_i - x_i^j \leq y_j \\ & \forall i = 1,\ldots,n,\ j = 1,\ldots,N \\ & Feas(x, A^0, b^0) \\ & Feas(x^j, A^j, b^j)\ \forall j = 0,\ldots,N \\ & x \in \mathcal{X}, y \in \mathbb{R}^{N}, z \in \mathbb{R} \\ & x^j \in \mathcal{X}\ \forall j = 1,\ldots,N \end{array}$

Contrary to RecOpt, there is no presolving step necessary. Depending on whether the median or center distance is minimized, and on the distance measure, we have the following size of the robust counterpart:

	#variables	#constraints
median, l_1	$n(2N+1)$	$N(n+m)+m$
median, l_∞	$n+N+nN$	$N(n+m)+m$
center, l_1	$n(2N+1)+1$	$N(n+m+1)+m$
center, l_∞	$n+N+nN+1$	$N(n+m+1)+m$

As it is the case for RecOpt, the user can choose whether to include nominal feasibility and variable constraints, or not.

6.4 Example Applications

In this section we present some basic ROPI functionalities on an example problem. Consider the following linear program given in fixed MPS format (taken from http://lpsolve.sourceforge.net/5.0/mps-format.htm):

```
NAME          TESTPROB
ROWS
 N  COST
 L  LIM1
 G  LIM2
 E  MYEQN
COLUMNS
    X         COST                1   LIM1                 1
    X         LIM2                1
    Y         COST                4   LIM1                 1
    Y         MYEQN              -1
    Z         COST                9   LIM2                 1
    Z         MYEQN               1
RHS
    RHS1      LIM1                5   LIM2                10
    RHS1      MYEQN               7
BOUNDS
 UP BND1      X                   4
 LO BND1      Y                  -1
 UP BND1      Y                   1
ENDATA
```

In ROPI, it is possible to read in both the fixed and free MPS format. We can read it in using simply

```
LP lp;
lp.read_mpsfile("file.mps");
```

to get the following LP:

$$\begin{aligned} \min\ & x + 4y + 9z \\ \text{s.t.}\ & x + y \quad \le 5 \\ & x \quad + z \ge 10 \\ & -y + z = 7 \\ & 0 \le x \le 4 \\ & -1 \le y \le 1 \\ & 0 \le z \\ & x, y, z \in \mathbb{R}. \end{aligned}$$

Now let us assume that some of the constraints are uncertain. We build an uncertainty set $\mathcal{U}$ consisting of two scenarios that randomly disturb the right-hand side of one constraint each. In ROPI, this can be achieved using

```
FiniteUncertainty unc;
```

```
list<Con>* lpcons = lp.get_cons();
for (int i=0; i<2; ++i)
{
  list<Con>::iterator it = lpcons->begin();
  advance(it,rand()%(lpcons->size()));
  FiniteScenario scen;
  scen.rhs[*it] = (it->rhs) * (1 + (rand()%100)/400.0 - 1.0/8 );
  unc.scenarios.push_back(scen);
}
```

The object `unc` now consists of two scenarios that modify the right-hand side of a constraint by up to 25%. Assume now that we randomly generate the scenario set $\mathcal{U} = \{(5, 10, 8), (5, 12, 7)\}$. The optimal objective value for the nominal case is then 54, while it is 62 and 80 in scenario 1 and 2, respectively.

We would like to solve the resulting uncertain optimization problem using the extended RecOpt counterpart with l_1 norm and center objective function. To do so, we create a robustness object that generates the required LP.

```
Robustness rob(&lp,ROB_RECOPT);
rob.set_uncertainty(unc);
RobustnessOptions opt;
opt.recobj = REC_CENTER;
opt.norm = NORM_L1;
opt.recopt_model = RECOPT_EXTENDED;
opt.solvertype = SOL_GUROBI;
rob.set_options(opt);
LP rc = rob.generate_robust();
```

Using only these couple of lines, we get the following robust counterpart for $\mathcal{U}$:

$$
\begin{array}{rrl}
 & \min & c \\
\text{s.t.} & \text{Nominal case} \left\{ \begin{array}{l} \\ \\ \\ \end{array} \right. & \begin{array}{l} x^0 - y^0 \leq 5 \\ x^0 + z^0 \geq 10 \\ -y^0 + z^0 = 7 \end{array} \\
 & \text{Scenario 1} \left\{ \begin{array}{l} \\ \\ \\ \end{array} \right. & \begin{array}{l} x^1 - y^1 \leq 5 \\ x^1 + z^1 \geq 10 \\ -y^1 + z^1 = 8 \end{array} \\
 & \text{Scenario 2} \left\{ \begin{array}{l} \\ \\ \\ \end{array} \right. & \begin{array}{l} x^2 - y^2 \leq 5 \\ x^2 + z^2 \geq 12 \\ -y^2 + z^2 = 7 \end{array} \\
 & \text{Optimality} \left\{ \begin{array}{l} \\ \\ \\ \end{array} \right. & \begin{array}{l} x^0 + 4y^0 + 9z^0 = 54 \\ x^1 + 4y^1 + 9z^1 = 62 \\ x^2 + 4y^2 + 9z^2 = 80 \end{array}
\end{array}
$$

$$\begin{aligned}
&\text{Nominal feasibility} \begin{cases} r_x - r_y \leq 5 \\ r_x + r_z \geq 10 \\ -r_y + r_z = 7 \end{cases} \\
&\text{Distance to nominal solution} \begin{cases} -a_x^0 \leq r_x - x^0 \leq a_x^0 \\ -a_y^0 \leq r_y - y^0 \leq a_y^0 \\ -a_z^0 \leq r_z - z^0 \leq a_z^0 \end{cases} \\
&\text{Distance to scenario 1 solution} \begin{cases} -a_x^1 \leq r_x - x^1 \leq a_x^1 \\ -a_y^1 \leq r_y - y^1 \leq a_y^1 \\ -a_z^1 \leq r_z - z^1 \leq a_z^1 \end{cases} \\
&\text{Distance to scenario 2 solution} \begin{cases} -a_x^2 \leq r_x - x^2 \leq a_x^2 \\ -a_y^2 \leq r_y - y^2 \leq a_y^2 \\ -a_z^2 \leq r_z - z^2 \leq a_z^2 \end{cases} \\
&\text{Objective constraints} \begin{cases} a_x^0 + a_y^0 + a_z^0 \leq c \\ a_z^1 + a_y^1 + a_z^1 \leq c \\ a_z^2 + a_y^2 + a_z^2 \leq c \end{cases} \\
&\text{Variable bounds} \begin{cases} 0 \leq x^0, x^1, x^2, r_x \leq 4 \\ -1 \leq y^0, y^1, y^2, r_y \leq 1 \\ 0 \leq y^0, z^1, z^2, r_z \end{cases} \\
&x^i, y^i, z^i \in \mathbb{R} \ \forall i \in \{0,1,2\} \\
&r_x, r_y, r_z \in \mathbb{R} \\
&a_x^i, a_y^i, a_z^i \in \mathbb{R} \ \forall i \in \{0,1,2\}
\end{aligned}$$

In a preprocessing step, all scenarios are solved to optimality - in this case, using Gurobi (determined by the `solvertype` option). The resulting robust counterpart is handed back to the user, who can then proceed to solve it using a `solver` object:

```
Solver sol;
sol.init(&rc, SOL_CPLEX);
sol.solve();
if (sol.get_statios() == SOL_OPTIMAL)
  sol.write_solition(cout,-1);
```

These lines generate a solver object, hand it the robust counterpart, and solve it using Cplex. If the problem is solved to optimality, the solution is printed to the standard output. In this example, the recovery robust solution turns out to be $(4, 0, 7)$ with a recovery distance of 2 to the three scenario solutions $(4, -1, 6)$, $(3, -1, 7)$, and $(4, 1, 8)$, while the optimal nominal solution would be $(4, -1, 6)$.

6.5 ROPI Extensions

Being an easily extensible library, ROPI still has room to grow. In this section, we would like to highlight some aspects that will play a major role in the further library development.

Further uncertainty types. As already described in Section 1.2, there are plenty of uncertainty types apart from finite uncertainty sets that are regularly used in the literature. However, consisting of infinitely many scenarios, they cannot be used to generate a generic robust counterpart directly, which is why the following aspect needs to be integrated in ROPI as well:

Scenario reduction. Even for finite uncertainties, robust counterparts often contain dominated constraints due to, e.g., worst-case scenarios. When constructing the robust counterpart in ROPI, this dominance can be checked on-the-fly. The applicability of reduction rules from infinite to finite uncertainty sets can be automatically checked, and if they should not be applicable, systematic sampling can be used. These reduction mechanisms are therefore related to the filters used in YALMIP.

Further robustness models. Furthermore, a direct extension is the implementation of further robust counterparts. Besides the already implemented concepts of strict and light robustness, recovery to optimality and recovery to feasibility, natural candidates would be the approach of Bertsimas and Sim or Mulvey et al, minmax regret optimization, or reliability.

Algorithmic plugins. The current philosophy of ROPI is to return a MIP robust counterpart to a given uncertain MIP. However, not all approaches to robust optimization presented in this work are based on integer programming techniques; further algorithmic approaches like the iterative procedure from Section 3.1.5.1, the sampling approach from Section 3.1.5.2, or algorithms from facility location to solve problems with non-linear distance measures can be added to the library.

7 Discussion and Outlook

Robustness has plentiful meanings in the English language. The first step of robust optimization, as in any mathematical discipline, is to find a definition in the strict language of mathematics that is free of ambiguity. Unfortunately, being still a recent field of research, neither has a shared definition of robustness been developed so far, nor a concise classification scheme. What all robustness concepts have in common, however, is that robustness is a beneficial property that is opposed to the nominal solution quality.

Therefore, the trade-off between robustness and nominal quality, i.e., the price of robustness, is crucial for the successful application of a robustness concept, where "successful" needs to be understood in the vague sense of "satisfying the needs of a real-world problem". Determining this trade-off from an analytical point of view leads to the problem that this is usually not the concept behavior we expect to see on real-world instances.

The purpose of this work was to bridge the gap between theory and practise in robust optimization. We hence claim that the methodology used is a natural approach to robustness. We introduced the versatile concepts of recovery-to-optimality and recovery-to-feasibility, and analyzed their behavior analytically. But every real-world problem has its particular requirements and uncertainty structure, and a generic concept needs to adapt to the circumstances. We therefore considered a wide spectrum of continuous and discrete problems, and discussed suitable robustness approaches. Table 7.1 shows an overview of all concepts and methods we applied to these problems.

We found that although the applicability of a robustness concept is determined by the uncertain problem, the methods we can use to solve a resulting robust counterpart depends on the computational complexity of the considered problem. Thus, a robustness concept can be seen as a framework, which is then turned into an algorithm that suits the concrete needs at hand.

We now discuss some further extensions of this work.

On the conceptual side, further research should focus on approaches that generalize both RecFeas and RecOpt. One way to do this is the following: Let an uncertain optimization problem $(P(\xi), \xi \in \mathcal{U})$ be given, and assume that the objective function f does not depend on ξ. As before, let $f^*(\hat{\xi})$ be the optimal objective value of the nominal scenarios. We assume that $f^*(\hat{\xi}) > 0$. Consider the problem

$$\begin{aligned}
&\min r \\
&\text{s.t. } d(x, \mathcal{F}(\xi, B)) \le r + \lambda(B) \ \forall \xi \in \mathcal{U}, B \in [0,1] \\
&\quad\ \ x \in \mathcal{X}
\end{aligned}$$

Problem Type	Problem	Concepts	Methods
Continuous	Linear programs (Netlib)	• Strict • RecOpt • RecFeas	• LP • iterative • sampling
Continuous	Aperiodic Timetabling	• Strict • Light • Recovery • RecOpt	• LP
Discrete	Loadplanning	• Strict • Bertsimas & Sim • Adjustable	• MIP
Discrete	Steiner Trees	• Strict • Adjustable	• MIP • approximation algorithms
Discrete	Periodic Timetabling	• Strict • RecOpt • Buffering	• MIP • local search
Discrete	Timetable Information	• Strict • Light	• dynamic programming

Table 7.1 Overview of experiments in this work.

where

$$\mathcal{F}(\xi, B) = \{x \in \mathcal{F}(\xi) : (1 - B)f(x) \leq f^*(\hat{\xi})\}$$

$\lambda(\cdot)$ denotes a function that relaxes the radius constraint depending on the nominal solution quality that is given by B. We have that $\mathcal{F}(\xi, 1) = \mathcal{F}(\xi)$ and $\mathcal{F}(\xi, 0) = \{x \in \mathcal{X} : x \text{ is optimal in scenario } \xi\}$.

This problem is a generalization of both RecFeas and RecOpt in the sense that we can model each concept with the right choice of λ. When setting

$$\lambda(B) = \begin{cases} 0 & \text{when } B = 1 \\ \max_{\xi^1, \xi^2 \in \mathcal{U}} d(\mathcal{F}(\xi^1), \mathcal{F}(\xi^2)) & \text{else} \end{cases}$$

the problem reduces to RecFeas, while it is equivalent to RecOpt when we set $\lambda(B) = 0 \; \forall B \in [0, 1]$. It seems promising to analyze how this concept behaves under different relaxation functions λ, and when it is still computationally tractable.

Another point of further research on RecFeas is the following iterative sampling algorithm that can be used for infinite uncertainty sets: We begin with an empty sample set $\emptyset = \mathcal{S}^{(0)} \subseteq \mathcal{U}$ and a starting solution $x^{(0)}$. In every iteration k, we determine a scenario $\xi^{(k)} \in \mathcal{U}$ that maximizes the distance $d(x^{(k-1)}, \mathcal{F}(\xi^{(k)}))$, and add it so the sample set $\mathcal{S}^{(k)} = \mathcal{S}^{(k-1)} \cup \{\xi^{(k)}\}$. We then determine a solution $x^{(k)}$ by solving RecFeas($\mathcal{S}^{(k)}$). If $x^{(k)} = x^{(k-1)}$, the algorithm has found an optimal solution to RecFeas($\mathcal{U}$).

Concerning the solvability of

$$\max_{\xi \in \mathcal{U}} d\left(x^{(k-1)}, \mathcal{F}(\xi)\right) = \max_{\xi \in \mathcal{U}} \min_{y \in \mathcal{F}(\xi)} d(x^{(k-1)}, y),$$

we can dualize the inner problem $\min_{y \in \mathcal{F}(\xi)} d(x^{(k-1)}, y)$ for fixed $x^{(k-1)}$ and ξ, if the recovery distance is polyhedral and and the uncertainty affine linear. For polytopic uncertainty sets, the problem of determining $\xi^{(k)}$ in every iteration is therefore solvable in polynomial time.

Also considering the problems we analyzed in this work, there are still many questions open:

Timetabling. Further research will focus on a systematic evaluation of different types of linear robustness functions; specifically, the robustness of a solution can be evaluated in a posteriori by generating a large set of delays, and solving the resulting delay management problem [SS10]. A necessary condition for a significant robustness function would then be that timetables with equal robustness have equal a posteriori robustness, irrespective of the travel time.

Loadplanning. The application of robustness models that give the planner a larger choice on how to react to parameter changes, as in the concept of recoverable robustness, is still an open issue.

Steiner Trees. A natural open problem for further research is to find constant-factor approximations to the one-stage stricty robust problem, and approximations to the two-stage strictly robust problem that further improve the bound of 15α that as developed in [DGRS05]. Another interesting aspect is to consider a two-stage concept that includes the recovery costs not only in terms of increased second-stage edge costs, i.e., we define a recovery distance measure $d(T^1, T^2)$ that gives the costs of changing a tree T^1 to a tree T^2. We are then able to use the results and methodology of RecFeas and RecOpt to find robust solutions that go beyond the concept of strict robustness.

Timetable Information. Again, further research includes to improve our algorithms and to apply other robustness concepts, such as recovery robustness to the problem of finding robust passenger paths. Here, a solution does not need to be feasible for all scenarios, but whatever is going to happen, we want to have a recovery algorithm at hand which is able to repair the solution if the scenario becomes known.

8 Summary of Contributions

This work is based on several papers, and therefore includes ideas of many coauthors. In this section we outline the author's contributions to each section.

Chapter 1 and 2

The lecture on robust optimization held by Anita Schöbel during the winter term 2009/2010 at the University of Göttingen is gratefully acknowledged. It was a great introduction to the topic and laid the foundation to my understanding of robustness.

Chapter 3

RecFeas. This Section is based on the joint work with Emilio Carrizosa, Mark Körner and Anita Schöbel [CGKS11]. Subsection 3.1.5 on iterative and sampling approaches was completely written by the author.

RecOpt. Some results of this section have been published in [GS11b] and [GS10] with Anita Schöbel.

Relations. The discussion of the relation between strict and adjustable robustness is based on a joint work with Anita Schöbel [GS12].

Chapter 4

Linear Programming. The experimental results on RecOpt were first presented in [GS11b]. The experiments were implemented and conducted by the author himself, and designed in joint work. The experiments on RecFeas are the author's sole work.

Aperiodic Timetabling. Parts of this Section have been published in [GS10]. While the theoretical work and the experimental design is coauthored with Anita Schöbel, all implementations and experiments were conducted by the author.

Chapter 5

Load planning. The section on robust loadplanning is a joint work with Sigrid Knust, Florian Bruns, and Anita Schöbel. In particular the experiments were conducted by Florian.

Steiner trees. This section makes use of discussions with Petra Mutzel, Markus Chimani, Bernd Zey, and Anita Schöbel. The proofs of the approximation ratios and the experiments were the author's work.

Periodic timetabling. Parts of this section on the local search heuristic have been published as [Goe12a]. The whole section is an individual work of the author.

Timetable information This section is based on the joint work [GKMH$^+$11] with Martin Knoth, Matthias Müller-Hannemann, Marie Schmidt and Anita Schöbel. In particular the experiments were conducted by Matthias and Martin.

Chapter 6

The library idea, architecture, functionality and all implementations were the author's sole work.

Bibliography

[ABV05] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack. In *Proceedings of the 13th annual European conference on Algorithms*, ESA'05, pages 862–873, Berlin, Heidelberg, 2005. Springer-Verlag.

[ABV09] H. Aissi, C. Bazgan, and D. Vanderpooten. Minmax and minmax regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427 – 438, 2009.

[AF90] J.-P. Aubin and H. Frankowska. *Set-valued analysis.* Systems and Control: Foundations and Applications, 2. Boston etc.: Birkhäuser. xix, 461 p., 1990.

[AL04] I. Averbakh and V. Lebedev. Interval data minmax regret network optimization problems. *Discrete Applied Mathematics*, 138:289–301, 2004.

[AT05] L. T. H. An and P. D. Tao. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of Operations Research*, 133:23–46, 2005.

[Ata06] A. Atamtürk. Strong formulations of robust mixed 0-1 programming. *Math. Program.*, 108(2):235–250, September 2006.

[AZ07] A. Atamtürk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. *Oper. Res.*, 55(4):662–673, July 2007.

[BC09] R. Blanquero and E. Carrizosa. Continuous location problems and big triangle small triangle. constructing better bounds. *Journal of Global Optimization*, 45:389–402, 2009.

[BCH09] R. Blanquero, E. Carrizosa, and P. Hansen. Locating objects in the plane using global optimization techniques. *Mathematics of Operations Research*, 34:837–858, 2009.

[BFJP12] N. Boysen, M. Fliedner, F. Jaehn, and E. Pesch. A survey on container processing in railway yards. *Transportation Science*, to appear, 2012.

[BGM10] A. Berger, M. Grimmer, and M. Müller-Hannemann. Fully dynamic speed-up techniques for multi-criteria shortest paths searches in time-dependent networks. In P. Festa, editor, *Proceedings of SEA 2010*, volume 6049 of *LNCS*, pages 35–46. Springer, Heidelberg, 2010.

[BGRS11] J. Byrka, F. Grandoni, T. Rothvoss, and L. Sanit. Steiner Tree Approximation via Iterative Randomized Rounding. *Journal of the ACM*, 2011. Invited publication.

[BK12] F. Bruns and S. Knust. Optimized load planning of trains in intermodal transportation. *OR Spectrum*, 34:511–533, 2012.

[BKK11] C. Büsing, A. M. C. A. Koster, and M. Kutschka. Recoverable robust knapsacks: the discrete scenario case. *Optimization Letters*, 5(3):379–392, 2011.

[BL97] J.R. Birge and F.V. Louveaux. *Introduction to Stochastic Programming.* Springer Verlag, New York, 1997.

[BMT04] Y. M. Bontekoning, C. Macharis, and J. J. Trip. Is a new applied transportation research field emerging? – A review of intermodal rail-truck freight transport literature. *Transportation Research Part A: Policy and Practice*, 38(1):1–34, 2004.

[BS03] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming Series B*, 98:2003, 2003.

[BS04] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[BT06] D. Bertsimas and A. Thiele. A robust optimization approach to inventory theory. *Operations Research*, 54(1):pp. 150–168, 2006.

[BTB08] A. Ben-Tal and O. Boni. Adjustable robust counterpart of conic quadratic problems. *Mathematical Methods of Operations Research*, 68:211–233, 2008.

[BTBB10] A. Ben-Tal, D. Bertsimas, and D. B. Brown. A soft robust model for optimization under ambiguity. *Oper. Res.*, 58(4-Part-2):1220–1234, July 2010.

[BTGGN03] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Math. Programming A*, 99:351–376, 2003.

[BTGN09] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization.* Princeton University Press, Princeton and Oxford, 2009.

[BTN98] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[BTN99] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25:1–13, 1999.

[BTN00] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Programming A*, 88:411–424, 2000.

[Büs09] C. Büsing. The exact subgraph recoverable robust shortest path problem. In Ravindra Ahuja, Rolf Möhring, and Christos Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 231–248. Springer Berlin / Heidelberg, 2009.

[Cal10] G. C. Calafiore. Random convex programs. *SIAM J. on Optimization*, 20(6):3427–3464, December 2010.

[CC05] G. Calafiore and M.C. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Math. Program.*, 102(1):25–46, January 2005.

[CC06] G.C. Calafiore and M.C. Campi. The scenario approach to robust control design. *Automatic Control, IEEE Transactions on*, 51(5):742 – 753, May 2006.

[CCG$^+$12] V. Cacchiani, A. Caprara, L. Galli, L. Kroon, G. Maroti, and P. Toth. Railway rolling stock planning: Robustness against large disruptions. *Transportation Science*, 46(2):217–232, May 2012.

[CDS$^+$07] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Robust Algorithms and Price of Robustness in Shunting Problems. In *Proc. of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS07)*, 2007.

[CDS$^+$09a] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Recoverable Robustness for Train Shunting Problems. *Algorthmic Operations Research*, 2009.

[CDS$^+$09b] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, A. Navarra, M. Schachtebeck, and A. Schöbel. Recoverable robustness in shunting and timetabling. In *Robust and Online Large-Scale Optimization*, number 5868 in Lecture Notes in Computer Science, pages 28–60. Springer, 2009.

[CDS$^+$09c] S. Cicerone, G. DAngelo, G. Stefano, D. Frigioni, and A. Navarra. Recoverable robust timetabling for single delay: Complexity and polynomial algorithms for special cases. *Journal of Combinatorial Optimization*, 18:229–257, 2009.

[CF02] E. Carrizosa and J. Fliege. Generalized goal programming: polynomial methods and applications. *Mathematical Programming*, 93(2):281–303, 2002.

[CG08] M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM J. on Optimization*, 19(3):1211–1230, November 2008.

[CGKS11] E. Carrizosa, M. Goerigk, M.-C. Körner, and A. Schöbel. Recovery to feasibility in robust optimization. Technical report, Preprint-Reihe, Institut fr Numerische und Angewandte Mathematik, Georg-August Universitt Gttingen, 2011. submitted.

[CGST08] A. Caprara, L. Galli, S. Stiller, and P. Toth. Recoverable-robust platforming by network buffering. Technical Report ARRIVAL-TR-0157, ARRIVAL Project, 2008.

[CKSW11] W. Cook, T. Koch, D. E. Steffy, and K. Wolter. An exact rational mixed-integer programming solver. In O. Günlük and G. J. Woeginger, editors, *IPCO 2011: Proceedings of the 15th International Conference on Integer Programming and Combinatoral Optimization*, volume 6655 of *Lecture Notes in Computer Science*, pages 104 – 116, 2011.

[CLSN11] D. Catanzaro, M. Labbé, and M. Salazar-Neumann. Reduction approaches for robust shortest path problems. *Comput. Oper. Res.*, 38(11):1610–1619, November 2011.

[CSSS08] S. Cicerone, G. Di Stefano, M. Schachtebeck, and A. Schöbel. Dynamic algorithms for recoverable robustness problems. In Matteo Fischetti and Peter Widmayer, editors, *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl Seminar proceedings, 2008.

[DDN09] G. D'Angelo, G. Di Stefano, and A. Navarra. Recoverable-robust timetables for trains on single-line corridors. In *Proceedings of the 3rd International Seminar on Railway Operations Modelling and Analysis - Engineering and Optimisation Approaches (RailZurich2009)*, 2009.

[DDNP11] G. D'Angelo, G. Di Stefano, A. Navarra, and C. M. Pinotti. Recoverable robust timetables: An algorithmic approach on trees. *IEEE Trans. Comput.*, 60(3):433–446, March 2011.

[Dem91] R. S. Dembo. Scenario optimization. *Ann. Oper. Res.*, 30(1-4):63–80, June 1991.

[DG12] J. Dongarra and E. Grosse. The Netlib LP benchmark set, August 2012. http://www.netlib.org/lp/.

[DGRS05] K. Dhamdhere, V. Goyal, R. Ravi, and M. Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, pages 367–378. IEEE Computer Society, 2005.

[DKSW01] Z. Drezner, K. Klamroth, A. Schöbel, and G.O. Wesolowsky. The Weber problem. In Z. Drezner and H.W. Hamacher, editors, *Facility Location: Applications and Theory*, pages 1–36. Springer, 2001.

[DMS08] Y. Disser, M. Müller-Hannemann, and M. Schnee. Multi-criteria shortest paths in time-dependent train networks. In C. C. McGeoch, editor, *WEA 2008. 7th International Workshop on Experimental Algorithms, Provincetown, MA, USA*, volume 5038 of *LNCS*, pages 347–361. Springer, Heidelberg, 2008.

[DSNP09] G. D'Angelo, G. Stefano, A. Navarra, and C. Pinotti. Recoverable robust timetables on trees. In D.-Z. Du, X. Hu, and P. Pardalos, editors, *Combinatorial Optimization and Applications*, volume 5573 of *Lecture Notes in Computer Science*, pages 451–462. Springer Berlin Heidelberg, 2009.

[Egg09] N. Eggenberg. *Combining robustness and recovery for airline schedules.* PhD thesis, EPFL, 2009.

[EMS09] A.L. Erera, J.C. Morales, and M. Svalesbergh. Robust optimization for empty repositioning problems. *Operations Research*, 57(2):468–483, 2009.

[ESB11] N. Eggenberg, M. Salani, and M. Bierlaire. Uncertainty feature optimization: An implicit paradigm for problems with noisy data. *Networks*, 57(3):270–284, 2011.

[Fal76] J. E. Falk. Exact solutions of inexact linear programs. *Operations Research*, 24(4):pp. 783–787, 1976.

[FM09] M. Fischetti and M. Monaci. Light robustness. In R. K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *Lecture Note on Computer Science*, pages 61–84. Springer, 2009.

[FSZ09] M. Fischetti, D. Salvagnin, and A. Zanette. Fast approaches to improve the robustness of a railway timetable. *Transportation Science*, 43:321–335, 2009.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[GKMH$^+$11] M. Goerigk, M. Knoth, M. Müller-Hannemann, M. Schmidt, and A. Schöbel. The Price of Robustness in Timetable Information. In Alberto Caprara and Spyros Kontogiannis, editors, *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 76–87, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[GM93] M. X. Goemans and Y.-S. Myung. A catalog of steiner tree formulations. *Networks*, 23(1):19–28, 1993.

[Goe12a] M. Goerigk. A local search algorithm for robust periodic timetabling. Technical report, Institut für Numerische und Angewandte Mathematik, 2012.

[Goe12b] M. Goerigk. *ROPI 0.1.0 Reference Manual*, June 2012. http://num.math.uni-goettingen.de/˜m.goerigk/ropi/.

[GS10] M. Goerigk and A. Schöbel. An empirical analysis of robustness concepts for timetabling. In Thomas Erlebach and Marco Lübbecke, editors, *Proceedings of ATMOS10*, volume 14 of *OpenAccess Series in Informatics (OASIcs)*, pages 100–113, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[GS11a] M. Goerigk and A. Schöbel. Engineering the modulo network simplex heuristic for the periodic timetabling problem. In P. M. Pardalos and S. Rebennack, editors, *Proc. of the 10th International Symposium on Experimental Algorithms (SEA 11)*, volume 6630 of *LNCS*, pages 181–192. Springer, 2011.

[GS11b] M. Goerigk and A. Schöbel. A scenario-based approach for robust linear optimization. In *Proceedings of the First international ICST conference on Theory and practice of algorithms in (computer) systems*, TAPAS'11, pages 139–150, Berlin, Heidelberg, 2011. Springer-Verlag.

[GS11c] J. Goh and M. Sim. Robust optimization made easy with rome. *Oper. Res.*, 59(4):973–985, July 2011.

[GS12] M. Goerigk and A. Schöbel. A note on the relation between strict robustness and adjustable robustness. Technical report, Institute for Numerical and Applied Mathematics, University of Göttingen, 2012.

[GSS] M. Goerigk, M. Schachtebeck, and A. Schöbel. LinTim - Integrated Optimization in Public Transportation. Homepage. see http://lintim.math.uni-goettingen.de/.

[GST12] K.-S. Goetzmann, S. Stiller, and C. Telha. Optimization over integers with robustness in cost and few constraints. In R. Solis-Oba and G. Persiano, editors, *Approximation and Online Algorithms (WAOA 2011)*, volume 7164 of *Lecture Notes in Computer Science*, pages 89–101. Springer Berlin / Heidelberg, 2012.

[Gur10] Gurobi Optimization, Inc., Houston, Texas. *Gurobi Optimizer Reference Manual Version 3.0*, September 2010.

[Gur12] Gurobi Optimization, Inc., Houston, Texas. *Gurobi Optimizer Reference Manual Version 5.0*, 2012.

[Hes12] M. Hess. Fahrplangestaltung mittels Matching-Verfahren. Master's thesis, Universität Göttingen, 2012.

[HUL93] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms.* Springer Verlag, Berlin, 1993.

[JL83] H. Juel and R. F. Love. Hull properties in locationproblems. *European Journal of Operational Research*, 12:262–265, 1983.

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KDV07] L. Kroon, R. Dekker, and M. Vromans. Cyclic railway timetabling: A stochastic optimization approach. In Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and Christos Zaroliagis, editors, *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 41–66. Springer Berlin / Heidelberg, 2007.

[KHA$^+$09] L. Kroon, D. Huisman, E. Abbink, P.-J. Fioole, M. Fischetti, G. Maróti, A. Schrijver, A. Steenbeek, and R. Ybema. The new dutch timetable: The or revolution. *Interfaces*, 39:6–17, 2009.

[KMV00] T. Koch, A. Martin, and S. Voß. SteinLib: An updated library on steiner tree problems in graphs. Technical Report ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin, 2000.

[KPY01] O.E. Karasan, M.C. Pinar, and H. Yaman. The robust shortest path problem with interval data. Technical report, Industrial Engineering Department, Bilkent University, Turkey, 2001.

[KW94] P. Kall and S.W. Wallace. *Stochastic Programming.* Wiley, Chichester, 1994.

[KY97] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications.* Kluwer Academic Publishers, 1997.

[KZ06a] A. Kasperski and P. Zielinski. The robust shortest path problem in series-parallel multidigraphs with interval data. *Operations Research Letters*, 34(1):69–76, 2006.

[KZ06b] A. Kasperski and P. Zieliski. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177 – 180, 2006.

[Lie06] C. Liebchen. *Periodic Timetable Optimization in Public Transport.* PhD thesis, Technische Universität Berlin, 2006.

[Lie08] C. Liebchen. The first optimized railway timetable in practice. *Transportation Science*, 42(4):420–435, 2008.

[LLMS09] C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *Lecture Note on Computer Science*, pages 1–27. Springer, 2009.

[LM07] C. Liebchen and R. Möhring. The modeling power of the periodic event scheduling problem: railway timetables — and beyond. In *Algorithmic Methods for Railway Optimization*, number 4359 in Lecture Notes on Computer Science, pages 3–40. Springer, 2007.

[Löf12] J. Löfberg. Automatic robust convex programming. *Optimization methods and software*, 27(1):115–129, 2012.

[LPW08] C. Liebchen, M. Proksch, and F.H. Wagner. Performances of algorithms for periodic timetable optimization. In *Computer-aided Systems in Public Transport*, pages 151–180. Springer, Heidelberg, 2008.

[Mar67] S. Marglin. *Public Investment Criteria.* MIT Press, Cambridge, 1967.

[MG05] R. Montemanni and L. M. Gambardella. The robust shortest path problem with interval data via benders decomposition. *4OR*, 3(4):315–328, 2005.

[MGD04] R. Montemanni, L. M. Gambardella, and A. V. Donati. A branch and bound algorithm for the robust shortest path problem with interval data. *Operations Research Letters*, 32(3):225–232, 2004.

[MHS10] M. Müller-Hannemann and S. Schirra, editors. *Algorithm Engineering — Bridging the gap between algorithm theory and practice*, volume 5971 of *LNCS*. Springer, Heidelberg, 2010.

[MS09] M. Müller-Hannemann and M. Schnee. Efficient timetable information in the presence of delays. In R. Ahuja, R.-H. Möhring, and C. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 249–272. Springer, Heidelberg, 2009.

[MSO06] M. Mani, A. K. Sing, and M. Orshansky. Joint design-time and post-silicon minimization of parametric yield loss using adjustable robust optimization. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, ICCAD '06, pages 19–26, New York, NY, USA, 2006. ACM.

[MSWZ07] M. Müller-Hannemann, F. Schulz, D. Wagner, and C. Zaroliagis. Timetable information: Models and algorithms. In *Algorithmic Methods for Railway Optimization*, volume 4395 of *LNCS*, pages 67–89. Springer, Heidelberg, 2007.

[MVZ95] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43(2):pp. 264–281, 1995.

[Nac98] K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables.* Habilitationsschrift, Deutsches Zentrum für Luft- und Raumfahrt Braunschweig, 1998.

[NO08] K. Nachtigall and J. Opitz. Solving periodic timetable optimisation problems by modulo simplex calculations. In *Proc. ATMOS*, 2008.

[NP09] S. Nickel and J. Puerto. *Location Theory - A Unified Approach.* Springer, 2009.

[Odi96] M. Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research (B)*, 30:455–464, 1996.

[Par12] Paragon Decision Company. *AIMMS - The Language Reference, Version 3.12*, March 2012.

[PC01] F. Plastria and E. Carrizosa. Gauge distances and median hyperplanes. *Journal of Optimization Theory and Applications*, 110(1):173–182, 2001.

[PC12] F. Plastria and E. Carrizosa. Minmax-distance approximation and separation problems: geometrical properties. *Mathematical Programming*, 132:153–177, 2012.

[Pee03] L. Peeters. *Cyclic Railway Timetabling Optimization.* PhD thesis, ERIM, Rotterdam School of Management, 2003.

[Pla84] F. Plastria. Localization in single facility location. *European Journal of Operational Research*, 18:215–219, 1984.

[RW91] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):pp. 119–147, 1991.

[RZ05] G. Robins and A. Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM J. Discret. Math.*, 19(1):122–134, May 2005.

[San09] P. Sanders. Algorithm Engineering – An Attempt at a Definition. In *Efficient Algorithms*, volume 5760 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2009.

[Sch09] M. Schnee. *Fully realistic multi-criteria timetable information systems.* PhD thesis, Fachbereich Informatik, Technische Universität Darmstadt, 2009. Published in 2010 by Südwestdeutscher Verlag für Hochschulschriften.

[Sch10] A. Schöbel. Light robustness and the trade-off between robustness and nominal quality. Technical report, Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Georg-August Universität Göttingen, 2010. submitted.

[SK10] Y. D. Sergeyev and D. E. Kvasov. *Lipschitz Global Optimization.* John Wiley & Sons, Inc., 2010.

[Soy73] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21:1154–1157, 1973.

[SS10] M. Schachtebeck and A. Schöbel. To wait or not to wait and who goes first? Delay management with priority decisions. *Transportation Science*, 44(3):307–321, 2010.

[Sti08] S. Stiller. *Extending concepts of reliability. Network creation games, real-time scheduling, and robust optimization.* PhD thesis, TU Berlin, 2008.

[SU89] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM J. Disc. Math.*, pages 550–581, 1989.

[SV08] R. Stahlbock and S. Voss. Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52, 2008.

[SY06] W. Sun and Y. Yuan. *Optimization theory and methods: nonlinear programming.* Springer, 2006.

[Ter09] T. Terry. *Robust Linear Optimization with Recourse: Solution Methods and Other Properties.* PhD thesis, University of Michigan, 2009.

[Thu80] D. J. Thuente. Duality theory for generalized linear programs with computational methods. *Operations Research*, 28(4):pp. 1005–1011, 1980.

[TSRB71] C. Toregas, R. Swain, C. ReVelle, and L. Bergman. The location of emergency facilities. *Operations Research*, 19:1363–1373, 1971.

[TTT08] A. Takeda, S. Taguchi, and R. Ttnc. Adjustable robust optimization models foranonlinear two-period system. *Journal of Optimization Theory and Applications*, 136:275–295, 2008.

[Wit64] C. Witzgall. Optimal location of a central facility: mathematical models and concepts. Technical Report 8388, National Bureau of Standards, 1964.

[WWR85] J. E. Ward, R. E. Wendell, and E. Richard. Using block norms for location modeling. *Operations Research*, 33:1074–1090, 1985.

[YKP01] H. Yaman, O. E. Karasan, and M. C. Pinar. The robust spanning tree problem with interval data. *Operations Research Letters*, 29:31–40, 2001.

[YKP07] H. Yaman, O. E. Karasan, and M. C. Pinar. Restricted robust uniform matroid maximization under interval uncertainty. *Math. Program.*, 110(2):431–441, March 2007.

[YY98] G. Yu and J. Yang. On the robust shortest path problem. *Computers and Operations Research*, 25:457–468, 1998.

[Zie04] P. Zielinski. The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research*, 158(3):570–576, 2004.

Curriculum Vitae

Persönliche Daten

Name	Marc André Goerigk
Geburtsdatum	18. Mai 1985
Geburtsort	Berlin
Staatsangehörigkeit	deutsch
Wohnsitz	Göttingen

Akademische Ausbildung

seit 11/2009	Wissenschaftlicher Mitarbeiter am Institut für Numerische und Angewandte Mathematik der Georg-August Universität Göttingen
	Promotionsstudium an der Fakultät für Mathematik und Informatik an der Georg-August Universität Göttingen
10/2005 - 10/2009	Studium der Mathematik an der Georg-August Universität Göttingen
	Abschluss: Diplom in Mathematik
06/2004	Abitur am Lise-Meitner Gymnasium Falkensee