Extending Concepts of Reliability

Network Creation Games, Real-time Scheduling, and Robust Optimization

vorgelegt von
Dipl.-Math. Sebastian Stiller, M. A.
Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Berichter:	Prof. Dr. Rolf H. Möhring
	Prof. Alberto Marchetti-Spaccamela
Vorsitzender:	Prof. Dr. Andreas Unterreiter

Tag der wissenschaftlichen Aussprache: 21. November 2008

Berlin 2009
D83

Meinen Eltern.

Acknowledgements

A lot of mathematical work must be done in solitude. Still, I conjecture that it is impossible to do mathematics without friends to inspire, share, and criticize one's own ideas. If this conjecture is true, then Berlin is a splendid city for mathematicians.

Berlin is an extraordinary place to work on applied and discrete mathematics. I had the pleasure to meet a number of excellent colleagues and great friends here. I would like to thank each one of them, and in particular those who were willing to be my co-authors. An additional special thanks goes to Axel Werner. Likewise, I benefited from the community of the ARRIVAL project, which an EU-reviewer once accurately called 'a circle of friends'.

It was Rolf Möhring who introduced me to this field of research, to the applications, and to the people. He provided me with everything a doctoral student could hope for. I could rely on his support when I needed it the most, and for this I am grateful to him. Rolf Möhring has a seemingly unshakeable trust in the virtue of passionate, free and self-responsible work. For a young academic this trust is demanding and encouraging at the same time—and sometimes it is a self-fulfilling prophecy.

Above all I would not be a mathematician without the company of my wife, but this is just a tiny detail among the many things I owe to Gabi.

CONTENTS

1

THREE CONCEPTS OF RELIABILITY

Apodictic certainty is the exceptional feature of mathematics among all sciences. Applying mathematical results in practice this certainty is lost. The theorems of applied mathematics are beyond doubt, but their application to real-world phenomena are necessarily argued by weaker means than logical deduction from axioms.

Strictly speaking arguments for the applicability of a certain mathematical theory to a certain real-world topic are never mathematical. But to a certain extent mathematical refinement can make the task of those non-mathematical arguments easier. The mathematical theory can incorporate a concept of *reliability*. This means that one acknowledges in the construction of the mathematical model a certain type of inevitable defect of the model compared to the modeled reality. This defect cannot be avoided, but one can formulate relevant statements that are true despite the specific, acknowledged defect. Thereby, the refined mathematical model instills a type of (still not apodictic) reliability to the application. One derives *reliable* statements from unreliable information.

In this work we consider three existing concepts for reliability from the areas of linear and combinatorial optimization, and discrete mathematics. For each of these concepts we achieve a significant enhancement.

A classical example is optimization under imperfect information. Here the defect lies in the fact that the available data for the optimization model is subject to uncertainty or changes. We do not know the exact data, but we know a set of possible scenarios. A classical concept of reliability in this setting is *robust optimization*. Robust optimization seeks to construct an optimal solution among those that are feasible in all likely scenarios. We develop the concept of *recoverable robustness*, which constructs an optimal solution among those that can be turned feasible in all likely scenarios by a limited recovery (cf. Chapter 4). This enhancement constitutes an important progress for both theory and practice.

For linear programming the notion of recoverable robustness enables to optimize with respect to more accurately shaped sets of likely scenarios. Here the basic idea is that in each scenario the total amount of data deviating from their reference value is restricted. So far one could impose such a restriction only to entries lying in the same inequality of the linear program (horizontal integration). Recoverable robustness (cf. Section 4.3) can consider restrictions to the deviation in the complete system (horizontal and vertical integration) of inequalities.

Introducing recovery to robustness allows for vertical integration preserving the compactness of the model and the quality guarantee for the solution. Further, vertical integration allows to consider non-trivial models for right-hand side disturbances, which are an important feature for many practical robust problems.

In practice classical robust optimization is not applicable to those real-world problems in which recovery is an essential aspect of the problem. Here recoverable robustness broadens the scope of applications for which robust solutions can be found. Among these new applications is delay resistant timetabling (cf. p. 104).

Finally, a polyhedral analysis of recoverable robustness for linear programs leads to a new perspective on linear programming under imperfect information (cf. Section 4.5). This perspective allows for a priori lower bounds on the probability of recovery robust solutions to be feasible in larger scenario set than those they are constructed for. These results are formulated independent from a distribution, allowing for specific bounds in case specific assumptions for the distribution are valid.

Recoverable robustness enhances a standard concept of reliability for the classical problem of optimization under imperfect information. Thereby, it enables robust solutions for large scope of real-world problems, and it opens the door for a better mathematical understanding of optimization under uncertainty.

A special concept of reliability with well established practical applications is that of a *feasibility test* for *real-time scheduling* (cf. Chapter 3). This is a striking example how mathematical progress can compensate for an inevitable defect in modeling.

An instance of real-time scheduling consists of an extremely large but highly structured set of jobs. The job sequence must be scheduled on a multiprocessor platform by a rule simple enough to be used in real-time. Typical applications like advanced radar systems, engine controls or autopilots are often safety critical. Therefore, the deadline of each job must be met. The task of a feasibility test is to determine in advance, whether the capacity of the platform and the scheduling policy suffice to meet all deadlines. For safety and reliability reasons one seeks for a mathematical guarantee. But there are two inherent defects. The first defect originates from the real-time character which inhibits the use of complicated scheduling policies. A second defect stems from the job sequence being too large (often even assumed countably infinite) to be considered explicitly. Only a compact description of the structure of the job sequence can be used for the feasibility test.

Moreover, in the case of *sporadic* real-time scheduling crucial data, namely, the release dates of the jobs is not fully determined by the compact description. Therefore, a sporadic instance can unfold in infinitely many different job sequences. The test should determine, whether each of these job sequences can be scheduled.

Since the nineties is has been conjectured that for a certain simple and widely used policy an approximate feasibility test in the following sense is possible: For any sporadic instance the test either recognizes it to be infeasible or to be feasible on a platform with processors of higher speed. Clearly, one seeks to keep the speed-up factor in such an approximate feasibility test as low as possible. By extending the standard concept of the load of an interval, we could eventually derive a such test. Moreover, this test has the lowest possible speed-up factor.

The first concept we discussed achieves reliability of solutions despite uncertain data. The second concept allows to assess the reliability of a real-time system although the complete data of that system cannot explicitly be part of the model. The third concept of reliability is the equilibrium of a certain type of game, a so-called *network creation game* (cf. Chapter 2).

A network creation game is a high-level tool of analysis for networks one cannot apprehend in full detail. Typical examples are social networks or the network of links between web-pages. Though there is an inevitable defect in exact knowledge about the network, one would like to gain some reliable, structural insight, e.g., the diameter, the degree sequence, or the so-called price of anarchy.

We analyze a network creation game in which several players establish a network of links among them. On the one hand, each player benefits globally from the topology of the network which connects the player directly or indirectly to each other player. On the other hand, each player has to pay locally for the topology, i.e., the number of her direct links. A player myopically, selfishly and uncooperatively maximizes her payoff, i.e., the difference between her benefit and cost.

What can be said about networks that are in equilibrium, i.e., stable in the sense that no single edge will be removed or added by the players? In particular, can we compare an anarchic network established by the players themselves to a network centrally designed in order to maximize the social payoff, i.e., the sum of all players' payoff? Remarkably, for the specific benefit we consider, one can rely on the players themselves to organize a network with no less than half the social payoff of a centrally planned network, in other words, the price of anarchy is 2.

So far the described type of network creation game could only be analyzed when the benefit of a player from another player depends linearly on their distance in the network. We are the first to analyze a network creation game with a benefit depending exponentially on the distance. This adds significantly to the games modeling power.

These three concepts of reliability all lie within the broader field of combinatorial and linear optimization, and discrete mathematics. Still, they belong to fairly disconnected special areas of research. We have to use significantly different mathematical tools to achieve each of the new results. Therefore, we present the three topics independently in three self-contained chapters, entitled by the broader area of the topic, Network Creation Games (Chapter 2), Real-time Scheduling (Chapter 3), and Recoverable Robustness (Chapter 4).

2

NETWORK CREATION GAMES

In this Chapter we study the following model: We are given a set of n vertices who want to communicate with each other. To this end they can establish links, i.e., edges among them. An edge causes a fixed cost c to both of its incident vertices. Yet, it can be used for communication by other vertices, too: For vertices v and w that have not established a direct edge between them, messages can be send via a common neighbor or any path connecting v and w. The disadvantage of indirect communication is the lower reliability of the transmission on a longer path. Each edge transmits the messages with an independent probability of δ. Thus, if v has shortest path distance to w equal to k, then the messages among them will reach their destination with probability equal to δ^k. For each vertex v the sum of the probabilities for successful communication with each other vertex is the benefit the vertex v gets from the network. This benefit minus the cost (c times the degree) is the payoff for that vertex from the network.

The network is constructed step by step by the vertices themselves. Therefore, the vertices are also called the players. In each step a pair of vertices decides, whether it wants to establish/maintain its common edge or not. The edge will be present immediately after the step, if and only if each of them has a better payoff with the edge than without it. So edges can be established or withdrawn.

Which networks are stable in the sense that no pair of vertices wants to withdraw its present edge or establish its non-present edge? This is the main question we answer in this chapter. In particular, we show which of the stable networks has the highest sum of payoffs over all vertices, and which is worst in this value. In fact, the ratio between the two is at most 2.

2.1 A Fundamental Graph Process

The type of object sketched in the motivation is called a network creation game, which is a special case of a graph process. A graph process is a sequence of graphs $(G_i)_{i\in\mathbb{N}}$, defined by a creation rule, which determines how G_{i+1} evolves from G_i (usually including some randomized decisions). If this creation rule can be formulated in terms of a game, we speak of a network creation game. Such objects are tools for high-level analysis of real-world networks.

Graph processes and network creation games help to understand the structure of real-world networks. Though these tools often fall short of a detailed modeling, their analysis elates by linking a simple and intuitive, creative principle to typical features of huge real-world networks. In this way, e.g., the preferential attachment model (cf. [16] for details) explains the scale-free structure formed by the pages of the WWW and their links.

Several such models have been proposed mostly in an economic context. We consider one which seems fundamental among these. The network created is a simple, undirected graph $G(V, E)$. It is created step by step. In each step a pair of vertices $\{u, v\}$ is chosen with no respect to whether their common edge already exists or not. For the edge to exist at the end of the step both vertices u and v have to benefit from it. In case at least one of them disapproves, the edge will not be present at the end of the step. A vertex benefits from an edge e, if the current graph with e gives that vertex a higher payoff than the current graph without e. The costs are *local*, namely every vertex pays a factor c times its degree, but the vertex enjoys income *globally*, namely from every other vertex exponentially declining with the distance to the other vertex. Whether to be at distance 2 or 3 is a much greater difference than whether to be at distance 100 or 101. These intuitions are not limited to economics. In the introductory motivation we consider the network as a means to send information from any vertex to any other vertex along paths. Unfortunately, each edge can have a probability of temporary failure of $(1 - \delta)$. Alternatively, at every edge used a $(1 - \delta)$-portion of the information sent into the edge is lost. Hence, every vertex v will send its deliveries to w via a shortest path and only a portion of $\delta^{\mathrm{dist}(v,w)}$ of the total amount sent from v to w and vice versa will reach its destination.

This creation rule can be understood as a game, where the vertices as players create the edges myopicly, selfishly, and non-cooperatively, though each edge requires both its end-vertices to agree. Obviously, it depends on the order of steps which networks are created. One is interested in certain equilibria or stable states, i.e., situations which every player would leave unchanged, if it was her turn now. The notion of stability suitable to this model is called *pairwise stability* (cf. [36]), i.e., a graph is stable, if it will stay unchanged, no matter which *pair* of vertices is chosen for the next step. Alternatively, one can define a probability distribution according to which the next pair of players is chosen. Then the game becomes a random graph process, i.e., a sequence of random variables $(G_i)_{i\in\mathbb{N}}$, each one representing a network. Again the same stable states and the possibility for the process to cycle are of primary interest. For the graph process, we accept every distribution that assigns a positive probability to every pair of vertices.

Besides the stable graphs, one is interested in graphs maximizing the sum of the payoffs, i.e., the total throughput of information minus the total edge cost. These graphs are called *efficient graphs* or *system optima*. The smallest ratio between the total payoff of a stable graph and that of a system optimum is called the *price of anarchy* of the graph process and is of high interest in network creation games since its first mentioning by Koutsoupias and Papadimitriou [39]. The classical notion of Nash-equilibrium (cf. [47]) is not adequate for bilateral games. Note that pairwise stability is a stricter notion of equilibrium as a player is not allowed to overhaul her whole strategy without the other players reacting to his steps (cf. also [27]).

Related Results The huge number of network creation games proposed in the literature shows the great interest for these explanatory tools. See Jackson [36] for a survey article. During the previous decade, also the interest in Nash equilibria and the price of anarchy for network creation games increased. The first to treat the price of anarchy were Koutsoupias and Papadimitriou in their seminal paper [39]. A relevant but simple network creation game is the unilateral game with linear payoff function as proposed by Fabrikant et al. [31]. Their model is the same as ours except for two features: For them an edge is used mutually, but only one of its end-vertices pays for it. Second, the income from other vertices decreases *linearly* with their distance. Recently, Albers et al. [8] give the best known upper bound on the price of anarchy for this model. Moreover, they disprove a structural conjecture for stable states made by Fabrikant et al. [31] when they show that graphs with cycles can be stable.

Corbo and Parkes [27] analyzed a bilateral consent-driven variant of the model by Fabrikant et al. and determined lower and upper bounds of the price of anarchy. Among other improvements Demaine et al. [28] lifted the lower bound to match the upper bound. Further, they compared these values with the unilateral model. As already stated by Corbo and Parkes, Nash equilibria are not appropriate for bilateral games. They therefore introduced the alleviated notion of *pairwise Nash* which is equivalent to pairwise stability.

The model we consider uses a more elaborate payoff function and the bilateral approach for sharing the costs of the edge. It was proposed by Jackson and Wolinsky [37] and interpreted as a graph process by Watts [58]. We state their results in Section 2.3. Though in comparison to those of [31] and [8] as well as [27] their results may appear limited to peculiar cases and immediate from the definition, the model of [37] and [58] is more convincing for two reasons: They give a consistent interpretation of mutual relations. And the income decreases exponentially with the distance.

Another model using bilateral cost sharing is given by Melendez-Jiminez [46]. Models using cost sharing principles are for example Bala and Goyal [15]. Anshelevich et al. [11] and [10] establish a near optimal solution for selfish players and determine the price of anarchy in a model with fair costs.

Our Contribution It would be desirable to achieve the same level of mathematical insight for the process of Jackson, Wolinsky [37] and Watts [58] as provided for the process of Fabrikant et al. [31] by Albers et al. [8], namely some structural knowledge of the stable states and bounds on the price of anarchy. We [19] achieve even more. Our process depends on two parameters, c and δ. First, we show that this process behaves equally whenever c is in $(\delta - \delta^2, \delta - \delta^3)$. For these cases we show that the process has positive probability to cycle.

Further, we provide for a thorough structural insight to stable states. On the one hand, this is of interest in its own. On the other hand, it allows for our main theorem: We give an explicit formula in c, δ, and the number of vertices for the exact price of anarchy for all $c \in (\delta - \delta^2, \delta - \delta^3)$. We argue by reducing the creation rule and payoff functions to local, graph theoretic criteria. In particular, the price of anarchy can be reduced to the number of edges in a maximum stable graph.

For $c > \delta - \delta^3$ we indicate how and to which extent our methods can be carried over. Further, we point out how an analogon to our main result is linked to extremal graph theory.

Overview of the Chapter In the next section we will collect some definitions and basic observations. The game we analyze behaves quite different for different choices of c and δ. In Section 2.3 we achieve some overview for this family of games. In the remainder we focus on one class of these games and analyze its stable graphs in several steps. Finally, we state two conjectures. At that time the reader will have an understanding, why we leave these as conjectures.

2.2 Preliminaries

For a graph G as usual $V(G)$ and $E(G)$ denotes its vertex respectively its edge set. For a pair of vertices $e = \{u, v\}$, we use $G+e$ and $G-e$, no matter whether $e \in E(G)$ or not, to denote the graph G with or without the edge e. The neighborhood of a vertex v will be denoted by $N(v)$, its degree by $d(v)$ $(= |N(v)|)$ and the distance between two vertices u and v by $\mathrm{dist}(u, v)$, i.e., the minimum number of edges in a path connecting u and v. If no path exists between u and v, we say $\mathrm{dist}(u, v) = \infty$.

Formally we define a *network creation game with exponential payoff* to be a triple of the cost coefficient, the income basis and the number of vertices, $(c, \delta, n) \in \mathbb{R} \times (0, 1) \times \mathbb{N}$.

For every vertex v the *income* is a function of $E(G)$ given by $\sum_{u \in V \setminus \{v\}} \delta^{\mathrm{dist}(u,v)}$. The *cost* of a vertex v is $c \cdot d(v)$, and its *payoff* is its income minus its cost.

Definition 2.1. *The* total (or social) payoff *is the sum of all vertices' payoff, i.e.,*

$$\sum_{v \in V} \sum_{u \in V \setminus \{v\}} \left(\delta^{\mathrm{dist}(u,v)} - c \cdot d(v) \right).$$

A graph maximizing the total payoff is called a system optimum.

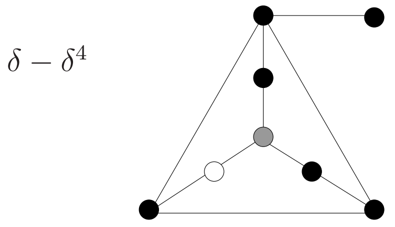
The following terminology is helpful for the classification of the games.

Definition 2.2. *A* situation *is a graph G together with a pair $\{u, v\}$ of its vertices.*

Every situation $(G, \{u, v\})$ defines by

$$\sum_{w \in V \setminus \{x\}} \delta^{\mathrm{dist}_{G+e}(x,w)} - \sum_{w \in V \setminus \{x\}} \delta^{\mathrm{dist}_{G-e}(x,w)}$$

a polynomial in δ for each of the pair's vertices ($x = u, v$) expressing the change in income (not yet in payoff) for that vertex between $G - e$ and $G + e$, with $e = \{u, v\}$. In these terms the creation rule reads as follows: When the (possible) edge $e = \{u, v\}$ is evaluated given the graph G, the *decision* will be positive, i.e., $G + e$ will be the resulting graph, if both polynomials at δ are greater than c, and negative, i.e., $G - e$ results, if at least one is less than c. Associating these polynomials to a situation will help to understand for which pairs of parameters δ and c the situation's decision is positive respectively negative. We do not consider cases where a polynomial can equal c, i.e., a vertex is indifferent about an edge. It would be possible to extend the model and results to these cases, but it would also be tedious.



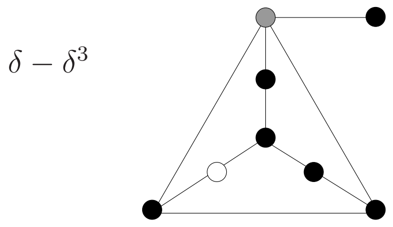


Figure 2.1: The polynomials for u (white vertex) in two situations on the same graph with different v (grey vertex).

Observe that if an edge is inserted by the game, this can only increase the total payoff. The deletion of an edge by the game can be locally advantageous but decrease the total payoff.

Now we define the central objects of interest, the stable graphs of a game.

Definition 2.3. *A graph G is called* stable*, if G together with any $e = \{u, v\} \in E(G)$ is a situation with positive decision, and G together with any $e = \{u, v\} \notin E(G)$ is a situation with negative decision.*

With this notion of stability we can define the Price of Anarchy for a games.

Definition 2.4. *The* price of anarchy *is defined as the maximum ratio of the total payoff of a system optimum over the total payoff of a graph G, over all stable graphs G:*

$$PoA(c, \delta, n) = \max_{G \text{ is stable for } (c,\delta,n)} \left(\frac{\textit{Total Payoff of a System Optimum}}{\textit{Total Payoff of } G} \right) .$$

2.3 A Classification of the Games?

Expressed in our terminology, Jackson and Wolinsky [37] and Watts [58] observe that for $c < \delta - \delta^2$ the complete graph is the only stable graph and the unique system optimum, because, no matter what the graph looks like, every further edge is beneficial. Trees have the least total cost among all connected graphs. In a star all not directly connected pairs of vertices are at distance 2. Therefore, if the cost factor c is high enough to draw at least some attention to the costs the star is optimal. Notably, this is the case for $c \in (\delta - \delta^2, \delta + \frac{n-2}{2}\delta^2)$. Beyond that limit for the costs, even the star's payoff becomes negative and the empty graph is the system optimum. Notably, the star is a stable graph for $c \in (\delta - \delta^2, \delta)$. Beyond that the empty graph is a stable state (though not the only one).

Of course the most interesting cases lie between the extremal games, where either the complete or the empty graph are stable. Still, these interesting cases are also more complicated. We will give some order for the set of those games.

Our first lemma links stability and the choice of c and δ to a structural property. The lemma gives a necessary condition for stability.

Lemma 2.5. *If a graph G is a stable state of a game with $c < \delta - \delta^{k+1}, k \in \mathbb{N}$, then G has diameter less than or equal to k.*

Proof. Assume to the contrary that there are two vertices u and v at distance greater than k. The (non-existing) edge $\{u, v\}$ would improve the income for u from at least v, i.e., the increase in income is greater than or equal to $\delta - \delta^{k+1}$. As the analogon holds for v, the edge would be inserted, which is to say, the graph is unstable. $\square$

Thresholds for c of the type $\delta - \delta^k$ seem to play an important role. We will now show that for small k they allow for a strong classification of games, notably, by the concept of identical games.

Definition 2.6. *A set of games $\{(c, \delta, n)\}$ is called* identical*, if and only if for every situation the decision is the same for all games in the set.*

Theorem 2.7. *For fixed n the set of games $\{(c, \delta, n) \mid c \in (\delta - \delta^2, \delta - \delta^3)\}$ is identical.*

Proof. Let G be the graph and $e = \{u, v\}$ be the edge of an arbitrary situation. If $\text{dist}_{G-e}(u, v) \geq 3$, then the income in $G + e$ is for u and for v at least $\delta - \delta^3$ higher than in $G - e$, because this is the minimal improvement in income from u for v and vice versa. As no distance gets longer by inserting an edge this lower bounds the total increase in income. As $c < \delta - \delta^3$ the change in payoff is positive.

Assume now $\text{dist}_{G-e}(u, v) = 2$ and the decision to be in favor of e. The gain of the edge for v from u (and vice versa) is $\delta - \delta^2$,i.e., less than its cost. Thus, for both u and v further vertices must be closer in $G+e$ than in $G-e$. For some $x \neq v$ *any* shortest path from u to x in $G + e$ must use the edge e, i.e., is of the form: $(u, v, \ldots, x)$. This implies that at least for one *neighbor* y of v all shortest path from u to y in $G + e$ go via v (and are shorter than those in $G - e$). We can conclude that the change in income for u from y is at least $\delta^2 - \delta^3$, as $\text{dist}_{G-e}(u, y) \geq 3$. Therefore, the total change in income for u is at least that from v plus that from y, so at least $\delta - \delta^3$. An analogon holds for v. Thus, a situation that is positive for one game with $c < \delta - \delta^3$ is positive for all of those.

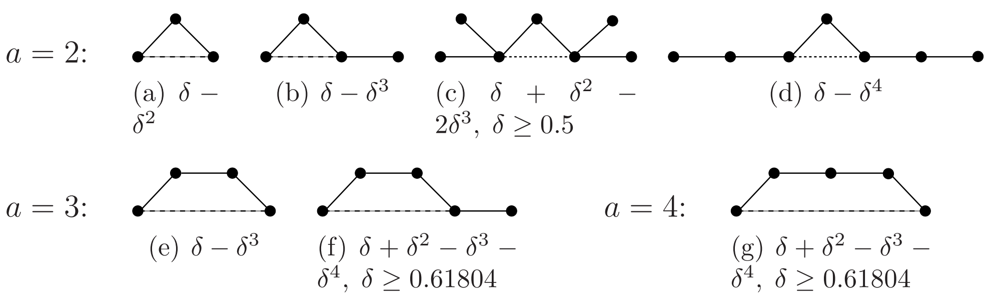
$\square$

As all games on n vertices are identical under the restriction $c \in (\delta - \delta^2, \delta - \delta^3)$, we also speak of *the* game. The argument of the theorem allows for all games with $c \in (\delta - \delta^2, \delta - \delta^3)$ to reduce a decision to a graph theoretical set of rules. An edge $e = \{u, v\}$ will be kept or inserted, if and only if at least one of the following conditions holds in the graph without e.

1. We have $\text{dist}_{G-e}(u, v) > 2$.
2. The end-vertex u has a neighbor x with $\text{dist}_{G-e}(v, x) = 3$, and the end-vertex v has a neighbor y with $\text{dist}_{G-e}(u, y) = 3$.

The model of Jackson, Wolinsky and Watts is rather a family of different models in their own right. The methods we used to analyze the case where $c \in (\delta - \delta^2, \delta - \delta^3)$ to a certain extent can be carried over to other cases.

Every situation gives rise to two polynomials in δ defined over the $(0, 1)$ interval and mapping to the positive reals determined by the change in income of the two vertices in question. We can also interpret the set $(0, 1) \times \mathbb{R}^+$ in which the graphs (here: graphs of functions) of those polynomials live as the set of all games for a fixed number of vertices n, because they are defined by a δ coordinate in $(0, 1)$ and a c coordinate in $\mathbb{R}^+$. In this picture a decision is positive for exactly those games that are below the polynomials of both vertices of the situation. This visualizes how the polynomials separate the set of all games in those for which the corresponding decision is positive and those for which it is negative. Restricting to $c \in (\delta - \delta^2, \delta - \delta^3)$ we exploit the nature of $\delta - \delta^2$ and $\delta - \delta^3$ as *threshold functions*. We call a function $f : (0, 1) \to \mathbb{R}^+$ a threshold function if for every polynomial p that stems



(a) $\delta - \delta^2$ (b) $\delta - \delta^3$ (c) $\delta + \delta^2 - 2\delta^3,\ \delta \geq 0.5$ (d) $\delta - \delta^4$

(e) $\delta - \delta^3$ (f) $\delta + \delta^2 - \delta^3 - \delta^4,\ \delta \geq 0.61804$ (g) $\delta + \delta^2 - \delta^3 - \delta^4,\ \delta \geq 0.61804$

Figure 2.2: Subnetworks that need to be considered in the proof of Theorem 2.8. The caption determines the benefit of the dashed edge. The lower bounds on δ given for some graphs guarantee that $c < \delta$ which makes the initial insertion of edges into an empty graph possible.

from a situation we have

$$\exists x_0 \in (0,1) : p(x_0) < f(x_0) \Rightarrow p(x) < f(x) \forall x \in (0,1)$$

and

$$\exists x_0 \in (0,1) : p(x_0) > f(x_0) \Rightarrow p(x) > f(x) \forall x \in (0,1).$$

One may carry on looking for threshold functions and redo our analyze for these cases. The next theorem gives the next threshold function.

Theorem 2.8. *For fixed n all games with $c \in (\delta - \delta^3, \delta - \delta^4)$ are identical.*

Proof. We show that a situation is either positive for all games in the interval or for none. Classify all situations by the distance $a := \operatorname{dist}(u,v)$ of the considered pair $\{u,v\}$. For $a \geq 4$ the increase in income is automatically bigger than $\delta - \delta^4 > c$, thus all decisions positive. For $a = 3$ the end-vertices u and v must attract each other with vertices not located on a shortest path between u and v to have a gain greater than $c > \delta - \delta^3$. Therefore they at least have an increase of $\delta - \delta^3 + \delta^2 + \delta^3$. This in turn always suffices. For $a = 2$ at least two further vertices outside the shortest paths are required for a positive decision. All positive situations for $c \in (\delta - \delta^3, \delta - \delta^4)$ contain Figure 2.2(d) or Figure 2.2(c) as subgraphs. The increase in income in Figure 2.2(d) is the smallest one and bigger than c.

$\square$

One may suppose that all polynomials stemming from situations are threshold functions. Yet, there are situations yielding polynomials that intersect each other in the open interval $(0,1)$. Consider the two situations depicted to the right in Figure 2.5. In both cases the black vertex has so many neighbors that the opposite vertex of the dashed edge will endorse the insertion of the dashed edge. The polynomial for the black vertex in the left situation minus that of the black vertex to the right gives: $\delta^3-\delta^5-(\delta^4-\delta^7)$. For some values of $\delta \in (0,1)$ this is negative, and for some it is positive. In other words, some games will insert the edge in the left situation but not in the right situation, whereas other games will in both situations do the contrary. This implies that for predicting the game's behavior it is no longer sufficient to specify one parameter by bounds of the other.

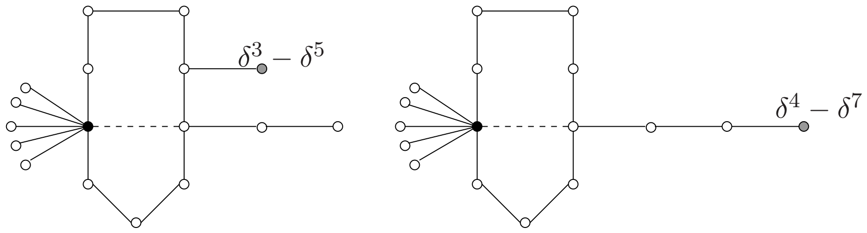


Figure 2.3: The insertion of the edge in question (dashed) strongly depends on the actual choice of c and δ. Cf. Figure 2.5 for the corresponding polynomials.

Some Classes of Games If the cost c is greater than δ, no edge can be inserted into the empty graph. So we concentrate on $c \in (0,1)$. Consider the unit square $(0,1)^2$, on the one hand, as the set of all games we are interested in, and on the other hand, as the space in which the graphs of the polynomials of all situations on n vertices live (Figure 2.4). The polynomials slice the square into areas of identical games. The two theorems of this section have shown that some of these slices are large open sets invariant for all n.

The first set of identical games is formed by those games that fulfill $c \leq \delta-\delta^2$, i.e., the games for which only the complete graph is stable, and in any situation the decision for the edge is positive. We call this set *Class 1*.

Class 2 comprises the games satisfying $\delta-\delta^2 < c < \delta-\delta^3$. This is the class, which we call 'the game' and which we will extensively analyze in the remainder.

We have seen in Theorem 2.8 that there is a *Class 3* comprising the games that satisfy $\delta-\delta^3 < c < \delta-\delta^4$. In principal, one may endeavor the same type of analysis as we do it here for Class 2. But, already the proof of Theorem 2.8 shows that this promises to be involved. We will see that the analysis of Class 2 is still non-trivial.

The example in Figure 2.5 shows that even the concept of classifying games by the notion of identical games will become clumsy.

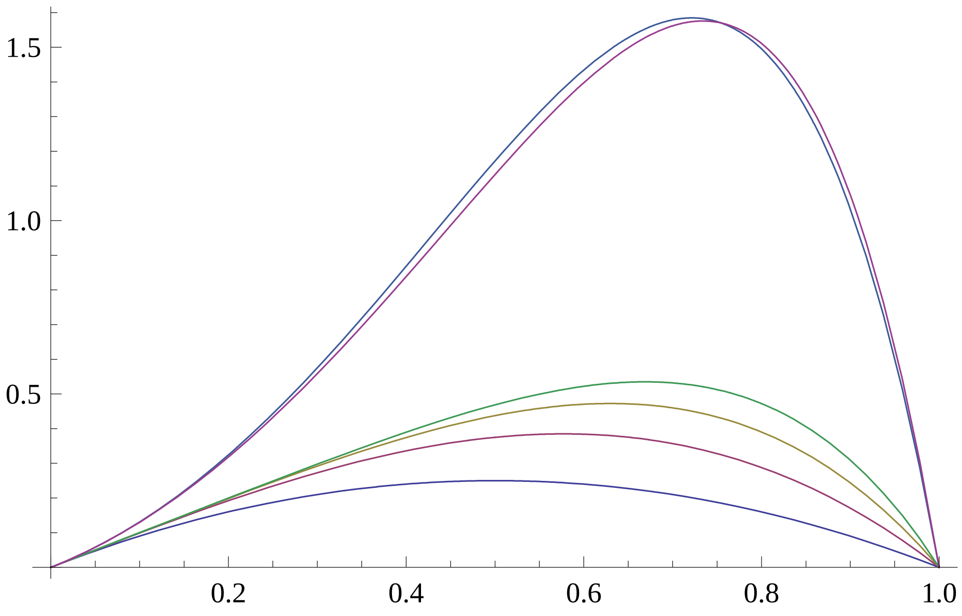


Figure 2.4: The polynomials of the situations in Figure 2.3, and those separating the classes 1, 2 and 3, and $\delta - \delta^5$.

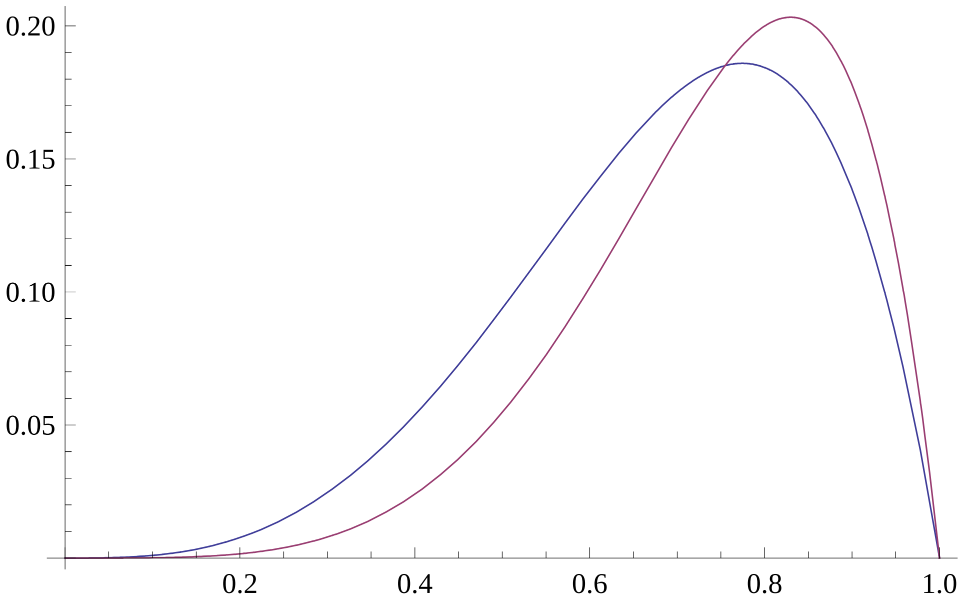


Figure 2.5: The polynomials of the situations in Figure 2.3 cross. Here we depict $\delta^3 - \delta^5$ and $\delta^4 - \delta^7$, i.e., both polynomials minus their identical terms $\delta + 3\delta^2 - 2\delta^4 - \delta^5 - \delta^6$.

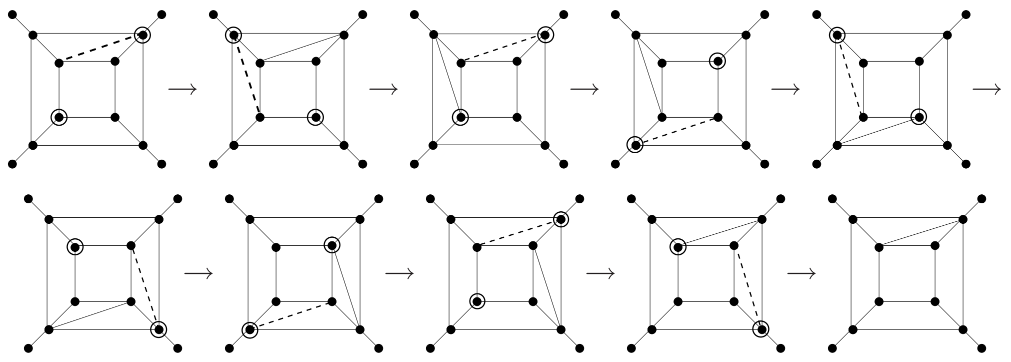


Figure 2.6: Cycling sequence of graphs. The dotted edge line connects the pair in question. The existence of other length 2 connections between the marked vertices decides whether the dotted edge is in or out.

2.4 Stability

In this section we get a first understanding about the game's stable graphs and their costs. In particular, we will get a lower bound on the price of anarchy. In Section 2.6 we will prove that this lower bound is the exact value for the price of anarchy.

Cycling The price of anarchy is defined with reference to the stable graphs. But there is an infinite series of pairs of vertices that never leads to a stable graph. Apply the graph rules above to the sequence depicted in Figure 2.6 to check that it cycles. In the first two situations the dashed edge is inserted because the two marked vertices are too far from each other. In the third situation the dashed edge is removed, because the two marked vertices have an alternative short connection.

Theorem 2.9. *The game for parameters c and δ with $c \in (\delta - \delta^2, \delta - \delta^3)$ can cycle.*

The Price of Anarchy In order to determine the price of anarchy, we need to establish criteria for stable graphs for the considered games.

For the game Lemma 2.5 amounts to say that every stable graph must have diameter exactly 2, as the complete graph obviously is not stable. Consequently, *the star is the only stable tree*. Further, we can give a sufficient condition for a graph to be stable.

Theorem 2.10. *If a graph G has diameter 2 and contains no triangles, then it is stable.*

Proof. For stability, on the one hand, we have to show that no edge in G will be removed. The graph G contains no triangles. Hence the shortest path between the end-vertices u and v of a currently present edge e in the graph without that edge, $G-e$, has length greater than or equal to 3. Thus, by the same argument as in Lemma 2.5, the edge is beneficial for both its endpoints and therefore kept.

On the other hand, no further edge will be inserted as the diameter suitable to the parameters is already reached: For any edge $e = \{u, v\}$ not present in G, calculating the payoff for one of its end-vertices u in $G + e$, is the same as in G except that the other end-vertex v will change from distance two to distance one. Consequently, the change in income by inserting e is exactly $\delta - \delta^2$ and therefore it is not beneficial to insert e.

□

By the above observations, we reduced the decisions of any situation to graph theoretic considerations. In fact, we can do the same for the price of anarchy, or equivalently, the total cost of a stable graph. For a stable graph we know all distances to be less than or equal to 2. Consequently, we can rewrite the total payoff of such a graph

$$\begin{aligned}\sum_{u\in V}\left(\sum_{v\in V\setminus\{u\}}\delta^{\mathrm{dist}(u,v)} - d(u)c\right) &= \sum_{\{u,v\}\in E}(\delta - c) + \sum_{\substack{\{u,v\}\in V\times V,\\ \mathrm{dist}(u,v)=2}}\delta^2 \\ &= m(\delta - c) + \left(\binom{n}{2} - m\right)\delta^2,\end{aligned}$$

where m denotes the number of edges. As $\delta - c - \delta^2 < 0$ by the choice of the parameters, it directly follows that the payoff of a stable graph is the bigger the less edges it has.

Lemma 2.11. *Let G and G' be stable graphs. The total payoff of G is greater than that of G' if and only if $|E(G)| < |E(G')|$.*

This together with the description of a stable graph in Theorem 2.10 provides for a lower bound on the price of anarchy. Recall that the star is the unique system optimum. It is well known that the graph $K_{\lfloor\frac{n}{2}\rfloor,\lceil\frac{n}{2}\rceil}$ maximizes the number of edges in a triangle free graph. For the lower bound of the price of anarchy as the ratio of this graph to the star, we need the stability of the graph $K_{\lfloor\frac{n}{2}\rfloor,\lceil\frac{n}{2}\rceil}$. But as the graph $K_{\lfloor\frac{n}{2}\rfloor,\lceil\frac{n}{2}\rceil}$ also has diameter equal to 2, we get from Theorem 2.10 that the maximum bipartite graph $K_{\lfloor\frac{n}{2}\rfloor,\lceil\frac{n}{2}\rceil}$ is stable.

Corollary 2.12. *The price of anarchy of the game with* n *vertices is bounded from below by*

$$\frac{(n-1)((\frac{n}{2}-1)\delta^2+\delta-c)}{\mu((\frac{n^2-n}{2\mu}-1)\delta^2+\delta-c)} \tag{2.1}$$

where $\mu := \lceil \frac{n}{2} \rceil \lfloor \frac{n}{2} \rfloor$ *is the number of edges in* $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$.

To show that Corollary 2.12 exactly states the price of anarchy for the game, we need to show that $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$ maximizes the number of edges among *all* stable graphs with n vertices. That would be easy if the converse of Theorem 2.10 was true, i.e., all stable graphs had no triangles. This is not the case as will be shown in the next part. There, we analyze the occurrence of triangles in detail. By the end of that part we will show (Theorem 2.18) that all stable graphs that contain at least one triangle have less edges than $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$. Therefore, we can conclude our main result:

Theorem 2.13. *For all games with* n *vertices and* $c \in (\delta - \delta^2, \delta - \delta^3)$*, the maximum price of anarchy equals* (2.1) *and is produced by the maximum bipartite graph* $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$ *against the star* $K_{1,n-1}$.

For $c \searrow (\delta - \delta^2)$ the expression (2.1), i.e., the price of anarchy, tends to 1 for every n and $\delta \in (0,1)$, whereas it converges to 2 for $c \nearrow (\delta - \delta^3)$, $n \nearrow \infty$, and $\delta \searrow 0$.

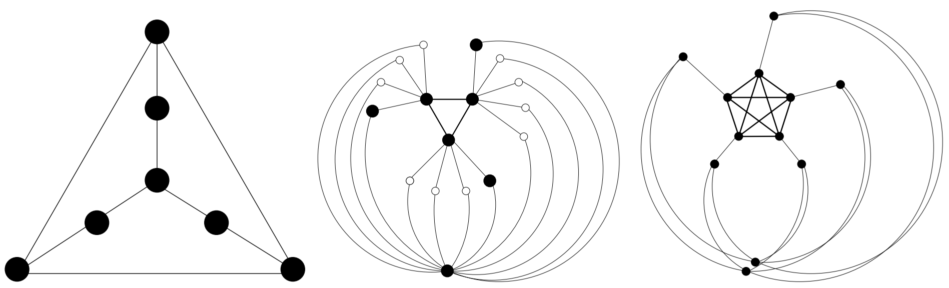


Figure 2.7: Stable graphs that contain a triangle.

2.5 Triangles

We have seen situations where a triangle is closed in the course of the game. There the game cycles and therefore does not reach a stable state. Nonetheless, there are stable graphs containing triangles. The three graphs depicted left in Figure 2.7 are stable and contain a triangle. The black vertices in the second graph form the leftmost graph. The white vertices in the second can be added one by one, such that for every number of vertices strictly greater than 6 stable graphs with at least one triangle are possible.

Here we deviate to show a structural result for stable graphs with triangles that is of interest in itself, though not necessary for the proof of our main result, Theorem 2.13.

Theorem 2.14. *If G is a stable state of the game and the vertex set $\{a, b, c\}$ forms a triangle in G, then there exists at least one $v \in V(G)$ with distances $\operatorname{dist}(a, v) = \operatorname{dist}(b, v) = \operatorname{dist}(c, v) = 2$.*

In order to show Theorem 2.14, we first need some lemmata.

Lemma 2.15. *Let G be stable and contain a triangle $\{a, b, c\}$. Then for every $i, j \in \{a, b, c\}$, $i \neq j$ there is a vertex in $N(i) \setminus \{a, b, c\}$ that has no edge to neither any vertex in $N(j)$ nor j itself.*

Proof. Assume the claim of the lemma is false. Then j can reach all neighbors of i via one of its own neighbors or directly. Moreover, it can reach i itself in two steps via the two other arcs of the triangle. That is as good as anything i can offer to j. Hence j would drop the arc $\{i, j\}$ contradicting the stability of G. □

Note that the premises of the following Lemmata 2.16 and 2.17 are assumptions to be falsified to prove Theorem 2.14. The proofs rest on the fact that without a vertex at distance 2 to all triangle vertices, those behave totally jealous towards their other neighbors.

Lemma 2.16. *Let G be stable, contain a triangle $\{a, b, c\}$, and have no vertex v with* $\operatorname{dist}(i, v) = 2$*, for all $i \in \{a, b, c\}$. Then the neighborhoods of the triangle vertices form a disjoint partition of $V(G) \setminus \{a, b, c\}$.*

Proof. First, it holds that $\bigcup_{i\in\{a,b,c\}} N(i) = V(G)$ because there is no vertex at distance 2 to the triangle and every stable graph has diameter less than 3. Assume vertex v to be in $N(i) \cap N(j)$, $j, i \in \{a, b, c\}, j \neq i, v \notin \{a, b, c\}$. Remove the arc $\{i, v\}$. One figures out quickly that i can still reach any vertex including v within two steps because even in G there is no vertex at distance 2 to all of the vertices a, b, c. Therefore, the assumption contradicts the stability of G. □

Lemma 2.17. *Let G be stable, contain a triangle $\{a, b, c\}$, and have no vertex v with* $\operatorname{dist}(i, v) = 2$*, for all $i \in \{a, b, c\}$. Then the neighborhoods of the triangle vertices are independent sets.*

Proof. Assume $v, w \in N(i) \setminus \{a, b, c\}$ for some $i \in \{a, b, c\}$ with $\{v, w\} \in E(G)$. When removing the edge $\{i, v\}$, considerations like in the previous proof show that the triangle vertex i can still reach every vertex within 2 steps. Hence G was not stable. □

Proof of Theorem 2.14. Assume the claim of the theorem not to be true for G, containing a triangle and being stable. Let x be a vertex in $N(a)$ as guaranteed to exist by Lemma 2.15 that has neither a connection to b nor to one of its neighbors, and y be a vertex in $N(b)$ that has neither a connection to c nor to one of its neighbors. As G is stable, its diameter is 2. Hence, x and y must have a common neighbor, as they cannot be adjacent by definition of x. By definition of y, such a neighbor is neither c nor one of c's neighbors. By Lemma 2.16 it is neither a nor b and by Lemma 2.17 it is neither a neighbor of b nor of a. Thus, we have a contradiction. □

Using Theorem 2.14 and the insights of Lemmata 2.15-2.17 we get that every stable graph that contains a triangle has at least 7 vertices. Further, the number of triangles in a stable graph does not need to be small, as the middle graph in Figure 2.7 shows, where the clique can consist of $\lceil \frac{n-1}{2} \rceil$ vertices. One figures out quickly that such a graph features the biggest clique in a stable graph of n vertices.

2.6 Stability Revisited

In order to determine the price of anarchy exactly, by Lemma 2.11 one has to look for the stable graph with the maximum number of edges. Intuitively, stable graphs with triangles should have more edges than triangle free stable graphs. We show that all stable graphs with triangles have less edges than some without triangles, namely the $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$.

Theorem 2.18. *For a number n of vertices, the maximum bipartite graph $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$ has the maximum number of edges among all stable graphs on n vertices.*

Proof. We need to show that $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$ has more edges than any stable graph on n vertices containing at least one triangle.

For every graph G we define a random variable based on the uniform distribution over the vertices of G as follows: Pick a vertex uniformly at random and sum up the degrees of its neighbors. Denote the expectation of that random variable by

$$\Phi(G) = \frac{1}{|V(G)|} \sum_{v \in V(G)} \sum_{u \in N(v)} d(u).$$

For short we write $\mu := \lfloor \frac{n}{2} \rfloor \cdot \lceil \frac{n}{2} \rceil$ for the number of edges in $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$. In order to show the statement of the theorem we prove two claims.

Claim 1: Let G be a stable graph with n vertices. Then

$$\Phi(G) \leq \Phi(K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}).$$

Claim 2: Let G be a graph with n vertices and μ edges. Then

$$\Phi(K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}) \leq \Phi(G).$$

These claims yield the statement of the theorem. Assume a stable graph G on n vertices with more edges than the maximum bipartite graph $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$. Arbitrarily remove edges from G until the resulting graph G' has exactly as many edges as $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$. Observe that removing an edge reduces the value of Φ by definition. Hence, $\Phi(G') < \Phi(G)$. By the second claim $\Phi(K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}) \leq \Phi(G')$. This implies $\Phi(K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}) < \Phi(G)$, which contradicts the first claim. Hence there is no stable graph with more than μ edges, which proves the theorem.

It remains to prove Claims 1 and 2. In addition, we will prove an even stronger version of the first claim, namely that every stable graph containing at least one triangle has a strictly smaller value of Φ than the complete bipartite graph and consequently less edges.

Proof of Claim 1 For the maximum bipartite graph we have $\Phi(K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}) = \mu$. Now, consider an arbitrary, stable graph G. For a randomly chosen vertex $v \in V(G)$ let $N(v)$ be all neighboring vertices and $b := |N(v)|$. Partition the edges incident to the vertices in $N(v)$ into three sets: first the edges incident to v, second those within $N(v)$ and third the edges to other vertices. The first and the last set together contain at most μ edges, because of the bipartiteness of the subgraph formed by these edges. Every edge in the second set belongs to a triangle containing v. Denote the number of vertices in $N(v)$ that belong to at least one triangle with v by ℓ. Then there are at most $\frac{\ell^2-\ell}{2}$ edges in the second set. As Φ counts the degrees of v's neighbors, each edge in the second set counts twice.

Assume a vertex u to be part of a triangle with v. Why will v be interested in the edge $\{u, v\}$? There must be at least one vertex that v can reach within two steps only via u, or $\text{dist}(u, v) \geq 3$ if the edge $\{u, v\}$ was removed. The latter is wrong, as u and v are in a triangle. Thus there exists a neighbor $w \neq v$ of u that is not connected to any vertex in $(N(v) \cup \{v\}) \setminus \{u\}$. In other words, u has an exclusive attraction to v in the sense that no other vertex in $N(v) \cup \{v\}$ is connected to w. Therefore we have to subtract $(b-1)$ from the number of possible edges in the third set for each of the vertices in $N(v)$ that participate in a triangle with v. Altogether, we get that the sum over the degrees of neighbors of v is at most

$$\sum_{u \in N(v)} d(u) \leq \mu + (\ell^2 - \ell) - \ell(b-1),$$

if ℓ neighbors of v participate in triangles with v. As $\ell \leq b$ and the preceeding inequality holds for all vertices $v \in V(G)$, we get

$$\Phi(G) \leq \mu. \tag{2.2}$$

This proves Claim 1 as stated above. Moreover, a graph containing a triangle does not achieve equality in Inequality (2.2). To show this, observe that in case of equality for each vertex the number ℓ must be in the set $\{0, b\}$, and if $\ell = b$ for a vertex v, then $N(v) \cup \{v\}$ is a clique. In other words, the neighborhood of a vertex v either contains no triangle with v or forms a clique together with the vertex v. For at least one vertex v with $d(v) > 1$ the latter must be true (otherwise G does not contain a triangle). A neighbor u of such a vertex v is not interested in its edge to v because u has a direct edge to all of v's neighbors and can reach v itself via another neighbor of v ($d(v) > 1$) within two steps. To report accurately, a graph G can fulfill at most two of the following three properties:

1. G is stable.
2. G achieves equality in Inequality (2.2).
3. G has a triangle.

Proof of Claim 2 Rewriting the counting function $\Phi(G)$ with

$$\sum_{v\in V(G)}\sum_{u\in N(v)} d(u) = \sum_{\{u,v\}\in E(G)} d(u) + d(v) = \sum_{v\in V(G)} d^2(v)$$

gives

$$\Phi(G) = \frac{1}{|V(G)|}\sum_{v\in V(G)} d^2(v).$$

Next we show that among all multiset of n natural numbers $s_i, 1 \leq i \leq n$ with $\sum_i s_i = 2m$ the degree sequence d_i of $K_{\lfloor \frac{n}{2}\rfloor,\lceil \frac{n}{2}\rceil}$ minimizes $\sum_i s_i^2$, which yields the claim of the theorem.

To see this consider any multiset of n natural numbers $s_i, 1 \leq i \leq n$ with $\sum_i s_i = 2m$. We show that the degree sequence d_i of $K_{\lfloor \frac{n}{2}\rfloor,\lceil \frac{n}{2}\rceil}$ minimizes $\sum_i s_i^2$ among all those multisets. Assume to the contrary that there is a multiset s^* distinct from d and minimizing the sum of squares. As it is distinct from d there exists a pair s_i^* and s_j^* with $|s_i^* - s_j^*| > 1$. W.o.l.g. we have $s_i^* > s_j^* + 1$ and can look at s', where $s_i' = s_i^* - 1$, $s_j' = s_j^* + 1$ and $s_k' = s_k^*\ \forall i \neq k \neq j$. We get $\sum_i {s_i'}^2 < \sum_i {s_i^*}^2$ contradicting the minimality of s^*. $\square$

2.7 Conjectures

We have analyzed the extremal stable graphs of Class 2. What can be said about the other games? Further, what happens if one plays the game at random, i.e., picks in every step a pair of vertices uniformly at random? Which graphs will usually result? What is their social payoff compared to that of the star?

We have seen in Section 2.3 that in general we cannot hope for such a clear situation as for Class 1 or 2. Nevertheless, general bounds for the price of anarchy might be achievable along the lines of this work. We conjecture the following:

Conjecture 2.19. *The price of anarchy is at most* 2 *for all games with* $c < \delta$.

Conjecture 2.20. *For* $\alpha < 1$ *there is a natural number* $n \in \mathbb{N}$*, such that with probability* 1 *a stable graph reached by any random game on* n *vertices has social payoff greater or equal to* α *times that of the star on* n *vertices.*

The are a number of reasons for these conjectures. First, recall that the social payoff of graph G with m edges in Class 2 is $C(G) = m(\delta - c) + \left(\binom{n}{2} - m\right)\delta^2$. Let S be the star, then $C(S)/C(G)$ tends to 1 with $n \to \infty$ as soon as m is subquadratic in terms of n. Thus, if we can show, that with probability 1 a stable outcome of the random game has only subquadratically many edges, we get Conjecture 2.20 for Class 2. (There are similar expressions for the social payoff in other games, classifying the vertex pairs by their distance.)

The first conjecture and the extension of the second to other games than Class 2 rest on an idea to show that Class 2 produces the greatest price of anarchy. When we concentrate on the number of edges, it is quite likely that games in which edges are expensive have stable states with at most as many edges as those games, in which edges are cheap.

This line of thought is supported by another rationale. Choose a game, where the star is optimal. Every stable graph will have diameter less than k if $c < \delta - \delta^{k-1}$. What can be said about graphs that have the maximum number of edges among all those containing no cycle smaller than $k-1$, i.e., in graph theoretic terms, which have *girth* $k-1$? They are not necessarily stable, but none of their present edges will currently be removed. The graph of our main result $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$ is an extremal graph in the sense that it maximizes the number of edges for girth 4. We conjecture that in general the graphs with maximum number of edges for girth $k-1$ are a good approximation for those maximizing the price of anarchy. Then the known upper bounds [9] for this long standing problem of extremal graph theory would imply that the price of anarchy becomes 1, as conjectured. For simplicity account for the price of anarchy as the greatest fraction between the number of edges in a stable and in an optimal graph. If our conjecture holds that the maximum graph of girth $k-1$ approximates the price of anarchy for game with $\delta - \delta^{k-1} < c < \delta - \delta^k$, this means that this price of anarchy would drop from worst ($\mathcal{O}(n)$) to best ($\mathcal{O}(1)$) as k growth. In other words, the users form an optimal network if the costs are very low ($c < \delta - \delta^2$) and an almost optimal network if the costs are very high. The worst outcome is then caused by costs $c \in (\delta - \delta^2, \delta - \delta^3)$, which is the case we have analyzed.

Difficulties Prima facie, these conjectures look like easy prey for a mathematician. For example the random game might be analyzable in the spirit of other random graph results. But, note that the random game after an initial phase, where it quite accurately resembles a standard random graph with a certain edge probability, later turns into random process controlled by heavily interdependent random variables. For example, in Class 2 an edge can only be removed, if it is part of a triangle. The analysis of triangle free random graphs (cf. [51] for a survey, and [30] for the path-breaking result) is already significantly more difficult than that of standard random graphs. There are many other obstacles for proving the conjectures. Still, we would be surprised, if they were not true.

3

REAL-TIME SCHEDULING

In the eighties an automotive manufacturer advertised its latest model to outclass the Gemini spaceships in on-board computational power. Today this comparison would embarrass even an ordinary mobile phone. The Gemini computing unit had a 20kB memory and could process 7,000 instructions per second. Despite the increased availability of computational power, one question remains topical: How much processing capacity is sufficient? How much processing capacity is sufficient to control the maneuvers of a spaceship or the fuel injection of an engine? Forty-three years after the on-board computer failed during landing of Gemini 4, we are able to answer this question.

At first glance an answer is not hard to find. The system has to run repeatedly a number of different types of calculations. The computational work for a calculation of a certain type is known or can be upper bounded. So one might sum up over all types how much computation is required on average per second. This load should be less than the available computational power per second. Such an utilization argument is indeed a very good answer to the question in special cases[1]. But in general, this is not sufficient, as at some intervals the load will be higher than on average. Note that the required calculations naturally have hard deadlines due to the physical process they are supposed to control. Thus we are not free to postpone a computation deliberately. To get a reliable upper bound for the required processing capacity it is necessary to take a closer look at the sequence of computation jobs and their scheduling.

One might be willing to design the space ship's computational unit slightly

[1]Here we refer to so-called implicit deadline systems, cf. p. 36.

or even substantially larger than needed to ensure that all calculations, which are necessary for the crew's save return, can be performed on time. Say, the computational power is more than twice the utilization. That seems to be enough to compensate for those subtle problems of scheduling. But how to be sure? One might even be prepared to invest a hundred times more computational power than needed. The point is, to be sure that one never needs a hundred and one times more. Understood the right way the gut-feeling, about double computational power sufficing, provably holds true. In this chapter we will explain the details.

The pertinent research area is called real-time scheduling. A real-time scheduling problem, as we investigate it, in principal constitutes a simple machine scheduling problem. We are a given a set of jobs. Each job has a processing time, a release date and a deadline. The jobs have to be scheduled on a certain number of (in out case) identical machines. Each job can be preempted. When the processing of the job is resumed one may choose to migrate the job to another machine. Still, we must not process a job on two machines at the same time. This seems to be an easy problem, and indeed as stated so far, a feasible solution can be found by a linear program[2] polynomial in the size of the input, i.e., the description of the jobs.

The size of the job's description is exactly where the matter becomes complicated. An instance of a *periodic real-time scheduling* problem confronts the scheduler with an infinite (countable) set of jobs, but specifies this set of jobs in a very compact, finite manner. Instead of describing each job separately, the instance is presented by a finite set of so-called *tasks*. A task emits infinitely many jobs of the same kind. All jobs of a task have the same processing time. Their relative deadline, i.e., the time span in which the job must be processed after its release, is also common to all jobs of one task. The release dates of all jobs of a common task are encoded by two values, an initial offset of the task and its period. When the offset time has elapsed the task emits its first job. After every release of a job, the task emits the next job period time units later. In this way, a countably infinite set of jobs is specified through a finite set of tasks, each described by four values.

Scheduling a task system means to schedule the jobs the tasks release with the additional requirement that at no point in time two jobs of the same task are scheduled in parallel. (It will be obvious that our results and methods can be applied even easier, in case this requirement is dropped.)

In fact, we will investigate a variant of the periodic real-time scheduling,

[2]The linear program formulation rests on a few preliminary insights. Notably, it is sufficient to specify the amount of work done for each job between two of an a priori known set of points in time. This set is comprised of the release dates and deadlines of the jobs, thus has linear size in the size of the input.

the *sporadic real-time scheduling*. For such problems we distinguish between an instance and its realizations. The instance consists of tasks as in the periodic case, except that the tasks lack an initial offset and the period is often called the minimal separation time. The period appears to be a separation time also in the periodic case, stating exactly how much time has elapsed between the release dates of two consequent jobs of the same task. From the specification of a *sporadic* instance we only know for each of its realization that the release times of any two jobs of the same task are separated by *at least* the tasks minimal separation time. Therefore, a sporadic instance encodes infinitely, in theory even uncountably many realizations. For sporadic instances we want a robust answer to the feasibility question. This means that we want to be sure that the scheduling is possible in all realizations of the sporadic instance.

Before we turn to the technical and exact presentation of the result, we will a guided tour to the ideas that underly our approach. Before we turn to this we take a glance at a typical example for this line of research and match them with our terminology. For further motivating examples and a general introduction we refer the interested reader to [24].

A modern engine features an elaborate control system for several of its functional parameters, like ignition timing or fuel injection. This control system consists of several tasks, each periodically or sporadically requiring a certain computation time. It is a task system with hard deadlines. There is no use for the correct injection parameter, if the injection is over. The goal for scheduling here is not to optimize but to find a feasible schedule.

Each task might emit a job several times per second. In comparison to the tiny time scale in which jobs repeat and must be finished, the total lifespan of an engine seems infinite by non-mathematical standards. And the mathematician will convince herself quickly that the lifespan is big enough that considering an infinite time horizon does not change the problem. So the decisive question when equipping the engine with a certain hardware to run the task system is, whether the hardware will be sufficient to meet all deadlines, or whether at some point in time it fails a deadline, which causes a malfunction of the engine. Therefore, in real-time scheduling we are looking for *feasibility tests*, i.e., for a procedure to test in advance, whether a task system can feasibly be scheduled on a certain hardware. Evidently, the application prohibits scheduling policies which themselves require extensive computation during the operation of the system. Simple policies like the earliest deadline first policy (EDF), least laxity first or even fixed priority strategies that prioritize the jobs according to a fixed ranking of the tasks are widespread in practice.

3.1 A Guided Tour to the Main Ideas

We want to convey a basic intuition for the difficulties in these questions, and for the line of attack, which we will follow. This line leads to our main result, the first tight constant approximate feasibility test for sporadic multiprocessor real-time scheduling.

A Frustrating Insight The most important insight for this chapter stems from a work on scheduling in general [50]. Imagine two companies with workforces equal in size. Both companies over time receive the same sequence of jobs, defined by release date, deadline, and processing requirement of each job. The first company is lead by an extraordinarily gifted manager with clairvoyant knowledge of the jobs that will occur, implementing a perfect scheduling of the jobs to the workers. The other company is lead by an old-school boss, who does not care about sophisticated scheduling. He makes sure of only two things: First, nobody is idle, unless there is no work left for him. Second, the boss pushes the employees to work twice as fast. The frustrating result of Lemma 2.6 in [50] is that for any moment in time, the second company will have at least as much work done as the first one.

A scheduling algorithm, which leaves no processor idle, whenever there is an available, unfinished job currently not being processed, is called a *busy* algorithm. The earliest deadline first (EDF) algorithm is a busy scheduling algorithm. By the lemma we get: If a sequence can be scheduled on a system at all, then it can be scheduled by EDF on a system with double speed processors.

It is easy to find some necessary conditions, whether an instance can be scheduled at all. But it is difficult to find a sufficient condition for scheduling in general, and also for scheduling by EDF. The mathematical value of this chapter lies in a sufficient and simple to check condition that EDF yields a feasible schedule on an instance with a little less than double-speed processors, which is a necessary condition for feasible scheduling by any algorithm on normal machines. So, for an instance the test either returns that it cannot be scheduled with the given platform of processors, or that it can be scheduled if each processor works almost twice as fast. The cited lemma indicates that such a test is possible. Extending a standard concept for analyzing real-time problems, namely the load of an interval, eventually yields this test.

Load Arguments and the Concept of Forced Demands The load of an interval Δ is a classical concept to achieve some understanding of what an optimal schedule must processes in an interval Δ. Some lower bound on the amount of work done by an optimal schedule in an interval is indispensable for our goal. The load of an interval is the ratio between its length, which indirectly characterizes the processing capacity available in the interval, and the demand in the interval. Classically, one counts as demand of Δ the sum of processing requirement of those jobs that are released and due within Δ. Ideally, one would like to count the amount of work *any feasible* schedule must process in Δ[3]. For both definitions of demand, the load in a feasible instance must be less or equal to the number of machines. For the feasibility test we define a demand, the so-called *forced forward demand*, which is stronger than the classical demand and close enough to the ideal demand, but still easy to compute a priori.

The classical definition of demand, which only considers jobs lying completely inside the interval, is too weak. Intervals may be filled up to capacity or beyond with jobs originating far earlier and being due far later. We are interested in the amount of work that must be done in an interval Δ by any algorithm. A job with release date shortly, say ϵ, before Δ and deadline shortly, say μ, after Δ, necessarily causes a demand equal to its processing requirement minus $\epsilon + \mu$ in Δ. Still, it is not counted in the classical load of Δ.

We want to count a part of a jobs processing requirement as necessary demand for an interval, even if the job is neither released nor in the interval. This will give a tighter bound to the amount of work that must be done in an interval Δ by any algorithm. It turned out that for our purposes it suffices to consider jobs with deadline *in* Δ and release date possibly before Δ. We count their processing requirement minus the distance of their release date to Δ as part of the necessary demand of the interval, the so-called *forced forward demand.*

Obviously, one can also define the *backward forced demand* or the *total forced demand* by the same idea. But the forward forced demand was sufficient for the feasibility test. Moreover, we give a fully polynomial approximation scheme for the maximal ratio between the forced forward demand and the interval length (i.e., indirectly the processing capacity of the processor platform) over all intervals in all realizations of a sporadic real-time scheduling instance.

Extra Speed or More Machines There is a valid objection against the old boss pushing his workers to double speed. Apart from the trade unions influence in the illustrative example it is a major point of criticism to this line of thought that higher speed processors are an uninteresting comparison. One has to use the processors which are currently available. When

[3]A similar point of view was recently adopted in [12].

constructing the processor platform for a hard real-time system it is little help to know, what would be possible, if the processors were faster.

Imagine we want to design a processor platform for a task system with the help of our test. Our test contains three criteria: one comparing the load (using forced forward demand) with the number of machines and two obvious necessary conditions, namely, that the processing time of each job is less than its relative deadline and less than its minimum separation time. We apply the test in the sense that we pretend to have processors with half the speed of the available ones. If the test ensures that EDF is feasible for double speed, than our actual platform will work for sure. If the test fails, we know that we cannot schedule it at all with slower processors. In case the failure is due to a violation of the load criterion, we can enlarge the number of processors on the platform until the criterion is fulfilled. But if a single job has processing time longer than its relative deadline or its minimal separation time, there is prima facie no way to change the platform design such that the system is feasible, unless one could upgrade to faster processors. As we run the test for half speed processors, the processing time of every job doubles. This means that jobs with a processing time of half their relative deadline (or minimum separation time) on the actual processors already cause the test to report infeasibility.

Vice versa, scheduling on double speed processors an instance deemed feasible for normal processors, means to assume processing times (on the fast machines) of at most half the relative deadlines and minimum separation times. This leaves enough slack for an EDF-schedule being feasible. Nevertheless, this is not completely trivial to show.

A comparison to a processor platform with more processors instead of higher speed processors would be more satisfying. In practice one may use more processors but one cannot use the processors of tomorrow. Actually, our result allows for a trade-off between extra speed and extra processors. Still, this trade-off is not a one-to-one exchange. For our test one must always assume some positive amount of extra speed.

Indeed extra speed is infinitely more powerful than extra machines: Consider a sequence of k jobs with relative deadline $2^i, 1 \leq i \leq k$, and processing time equal to the relative deadlines. If the jobs arrive such that all of them are present for at least one point in time, then k single speed machines are necessary. Yet, the instance can always be scheduled on one double speed machine by EDF.

The Least Possible Speed-up The solution to a real-time scheduling problem is a pair of a simple scheduling policy and a (approximate) feasibility test that recognizes which instances can be scheduled by the chosen policy and which are infeasible under that policy. Thereby, the solution makes a double error in comparison to an optimal clairvoyant scheduling. The first error occurs, because the simple scheduling policy cannot schedule all instances that are feasible under optimal scheduling. The second error occurs, because the feasibility test cannot identify exactly those instances that can be scheduled by the simple policy.

The result of [50] can be interpreted as an upper bound to the first error: Measure the first error by the speed-up factor σ such that the simple scheduling policy EDF can schedule on σ-speed machines every instance that can be scheduled on single-speed machines by an optimal policy. The result of [50] shows that this σ for EDF is not greater than $2 - 1/m$. In fact, they also show that the σ for EDF cannot be less than this. In other words, the bound for the first error is tight.

This lower bound to the first error of EDF can be derived in the following way. For a platform of m machines consider a job sequence of one job with processing requirement equal to 1 and a large number of sufficiently small jobs with total processing requirement equal to $m - 1$. The release of all jobs is zero and the deadline equal to 1 except for the large jobs that has a slightly later deadline. EDF will finish all small jobs before it starts the large one. Given a platform with σ-speed machines the large job start no earlier than at time $(m - 1)/(\sigma m)$. It is easy to see that σ must be at least $2 - 1/m$ for the large job to finish in time.

The extra speed required by the test we devise is $2 - 1/m + \epsilon$. Actually, we give a criterion for feasibility on $(2 - 1/m)$-speed machines. A further infinitesimal extra speed is necessary, because we evaluate that criterion by an FPTAS.

Some first error is unavoidable, because real-time scheduling requires simple policies. For EDF the first error is $2 - 1/m$. Our test only adds an infinitesimal second error.

3.2 Related Work

We study the problem of scheduling recurring processes, or tasks, on a multiprocessor platform. An instance of the problem is given by a finite set I of tasks, which need to be executed by the system; each task generates a possibly infinite sequence of jobs. In the following we denote by n the cardinality of I.

In the *periodic* version of the problem, a task τ, $\tau \in I$, is characterized by a quadruple of positive numbers: an offset o_τ that represents the time instant when the first job generated by the task is released, a processing time c_τ, a relative deadline D_τ and a period T_τ. Each occurrence of task τ is represented by a job: the k-th occurrence of task τ is released at time $o_\tau + (k-1)T_\tau$, requires at most c_τ units of processor time and must complete its execution before time $o_\tau + (k-1)T_\tau + D_\tau$. Note that a task defines an infinite sequence of jobs, but a given set of tasks generates exactly one job sequence.

In the *sporadic* case, each task is characterized by a triple (c_τ, D_τ, T_τ) where c_τ, D_τ have the same meaning as in the periodic case, while T_τ denotes the *minimum* time interval between successive occurrences of the task. Note that in a sporadic task system the time instant when the next invocation of a task will be released after the minimal separation time has elapsed is unknown. Therefore, a given set of tasks can generate infinitely many sequences of jobs.

The correctness of a hard-real-time system requires that all jobs complete by their deadlines. A periodic (sporadic) task system is *feasible* if there is a feasible schedule for any possible sequence of jobs that is consistent with the period, deadline, and worst-case execution time constraints of the task system, and it is *schedulable* by a given algorithm if the algorithm finds a feasible schedule for every such sequence of jobs. In the sequel we focus on preemptive scheduling algorithms that are allowed to interrupt the execution of a job and resume it later.

Given a scheduling algorithm A, a *schedulability test* for A is an algorithm that takes as input a description of a task system and answers whether the system is schedulable by A or not. A schedulability test is *exact* if it correctly identifies all schedulable and unschedulable task systems and it is *sufficient* if it correctly identifies all unschedulable task systems, but may give a wrong answer for schedulable task systems. A sufficient schedulability test that can verify whether a given job set is schedulable is a natural requirement for a scheduling algorithm that must be used in hard-deadline real-time applications. In fact, from a practical point of view, there is no difference between a task system that is not schedulable and one that cannot be proven to be schedulable.

In the case of a single machine, the problem has been widely studied and effective scheduling algorithms are well understood [18, 45]. In this paper we study scheduling algorithms for sporadic task systems on parallel machines. The problem is not only interesting from a theoretical point of view but is also relevant in practice. In fact, real-time multiprocessor systems are becoming common: there are single-chip architectures, characterized by a small number of processors and large-scale signal-processing systems with many processing units.

There is an extensive literature on real-time scheduling. We limit the following discussion to the results that are more relevant to our work.

Single Machine Scheduling In the case of a single machine it is known [18, 29, 45] that the earliest deadline first scheduling algorithm (EDF), which at each instant in time schedules the available job with the smallest deadline (with ties broken arbitrarily), is an optimal scheduling algorithm for scheduling a periodic (or sporadic) task system in the following sense: if it is possible to preemptively schedule a given collection of independent jobs such that all the jobs meet their deadlines, then the schedule generated by EDF for this collection of jobs will meet all deadlines as well.

Despite this positive result, we remark that the feasibility test for periodic task systems, although solvable in exponential time, is strongly co-NP-hard even in special cases [18, 41].

Approximate feasibility tests have been proposed that allow the design of efficient feasibility tests (e.g. running in polynomial time) while introducing a small error in the decision process that is controlled by an accuracy parameter. Such approaches have been developed for EDF scheduling and for other scheduling algorithms.

Two different paradigms can be used to define approximate feasibility tests: pessimistic and optimistic. If a pessimistic feasibility test returns "feasible", then the task set is guaranteed to be feasible. If the test returns "infeasible", the task set is guaranteed to be infeasible on a slower processor, of computing capacity $(1-\epsilon)$, where ϵ denotes the approximation guaranteed.

If an optimistic test returns "feasible", then the task set is guaranteed to be feasible on a $(1+\epsilon)$-speed processor. If the test returns "infeasible", the task set is guaranteed to be infeasible on a unit-speed processor [26].

Fully polynomial-time approximation schemes (FPTAS) are known for a single processor; in fact for any $\epsilon > 0$ there exists a feasibility test that returns an ϵ-approximation; the running time of the algorithm is polynomial in the number of tasks and in $1/\epsilon$ (see for example [6, 7, 26, 32] and references therein).

Finally we observe that, in the case of one processor, the sporadic feasibility problem is known to reduce to a special case of the periodic problem, where all tasks have offset 0 (i.e. each task releases its first job at time zero).

Multiple Machine Scheduling We first observe that in the multiprocessor case the previous analogy between sporadic and periodic problems is not true.

Regarding the analysis of EDF, it is known [50] that any *feasible* task system on m machines of unit capacity is EDF-schedulable on m machines of speed $2 - 1/m$. This result holds for EDF and every other so-called busy policy. In a busy policy no machine is idle unless all currently available jobs are being processed. Subsequent work has analyzed the advantage of trading speed for machines [40], while further work on conditions for the schedulability of EDF has been done by Baker [13].

Note that the result of [50] *does not* imply an efficient test for deciding when EDF (possibly with extra speed) can schedule a sporadic task system. Thus, the main open problem in order to apply the result of Phillips et al. [50] is the lack of a feasibility test.

The problem has attracted a lot of attention in recent years (see e.g. [14] and references therein for a thorough presentation). A number of special cases have also been studied; for example, when for each task the deadline is equal to the period (*implicit-deadline* task systems), it has been shown that

$$\sum_{\tau \in I} \frac{c_\tau}{T_\tau} \le m \text{ and } \max_{\tau \in I} \frac{c_\tau}{T_\tau} \le 1$$

gives a necessary and sufficient test for feasibility of the system.

However, not much was known regarding the feasibility of an arbitrary-deadline task system. A sufficient test in this case is given by

$$\sum_{\tau \in I} \frac{c_\tau}{\min(D_\tau, T_\tau)} \le m \text{ and } \max_{\tau \in I} \frac{c_\tau}{\min(D_\tau, T_\tau)} \le 1,$$

but this test is far from approximating a necessary condition, i.e., it does not provide a good approximate feasibility test in general (it is not hard to see that there exist feasible task systems for which $\sum_{\tau \in I} c_\tau / \min(D_\tau, T_\tau)$ can be $\Omega(m \log m)$).

To the best of our knowledge, no better bound is known. We refer the reader to the survey [14] for feasibility tests that are known for other special cases.

Our Contribution We [23] give the first constant-approximate feasibility test for sporadic multiprocessor real-time scheduling.

Namely, we give a test that, given a sporadic multiprocessor instance I, decides whether it can be scheduled by EDF on m speed-$(2 - 1/m + \epsilon)$ machines, or shows that the instance violates at least one of three basic conditions, which are necessary for schedulability on m speed-1-machines. In fact we give a slightly stronger result, allowing to trade some extra speed for extra machines. Note that in general extra machines are less powerful than extra speed.

Two of the basic conditions are trivial. The third condition provides a lower bound on the processing requirement of an interval. We call it the *forward forced demand*. This concept is strong enough to approximately capture the feasibility of scheduling a sporadic task system on a multiprocessor platform; however it is simple enough to be approximated in polynomial time up to an arbitrarily small $\epsilon > 0$: in Section 3.5 we give an algorithm that checks the third condition in time polynomial in the input size of I and $1/\epsilon$, for any desired error bound $\epsilon > 0$.

In the meantime, independently a different test with a worse speed-up factor of 2.618 has been proposed [17].

3.3 The Model

An instance I is a finite set of tasks. Each task $\tau \in I$ is a triple of positive numbers, namely, a processing time c_τ, a relative deadline D_τ and a period or minimal separation time T_τ. Every job j belongs to a task τ_j, and has a release date $r_j \geq 0$. We write $c_j := c_{j_\tau}$, and $D_j := D_{j_\tau}$, and $T_j := T_{j_\tau}$, and we call $d_j := r_j + D_j$ the (absolute) deadline of j. We assume D_τ, c_τ, $T_\tau \in \mathbb{N}$.

A (sporadic) *job sequence* R of an instance I is an arbitrary, countable set of jobs, all belonging to tasks in I, with the following property: Any pair of distinct jobs j and k belonging to the same task τ satisfies $|r_j - r_k| \geq T_\tau$.

Formally a feasible schedule for a job sequence R on m machines is a set of measurable functions $S_j : \mathbb{R}^+ \to \{0, \ldots, m\}$, one function for each $j \in R$ satisfying the following conditions. (Interpret $S_j(t)$ as the index of the machine on which job j is scheduled at time t, unless $S_j(t) = 0$, which means that job j is not scheduled at time t.)

- Everything is scheduled: $\forall j \in R : c_j = \sum_{p=1}^{m} |S_j^{-1}(p)|$.
- Deadlines and release dates are respected: $\forall j \in R : \bigcup_{p=1}^{m} S_j^{-1}(p) \subseteq [r_j, d_j]$.
- Each machine processes at most one job at a time: $\forall p \in \{1, \ldots, m\} : \forall j \neq g \in R : S_j^{-1}(p) \cap S_g^{-1}(p) = \emptyset$.
- Jobs of the same task are not scheduled in parallel:
$$\forall j \neq g \in R : \tau_j = \tau_g \Rightarrow \bigcup_{p=1}^{m} S_j^{-1}(p) \ \cap \ \bigcup_{p=1}^{m} S_g^{-1}(p) = \emptyset.$$
- No job is processed by two machines at the same time:
$$\forall j \in R, \forall p \neq q \in \{1, \ldots, m\} : S_j^{-1}(p) \cap S_j^{-1}(q) = \emptyset.$$

Preemption and migration of jobs are explicitly allowed.
Given a real number x we denote by x^+ its positive part, that is $x^+ := \max(x, 0)$.

3.4 A Feasibility Test

Definition 3.1. *Consider a job j with release date r_j, absolute deadline d_j, and processing time c_j satisfying $d_j \geq r_j + c_j$ (i.e., for its task we have $D_{\tau_j} \geq c_{\tau_j}$). For a non-empty interval $\Delta = [t, t')$ with $d_j \in \Delta$, we call*

$$f(j, \Delta) := \left(c_j - (t - r_j)^+\right)^+$$

the forward forced demand *of j in Δ.*

Note that, for a job j and an interval Δ such that both deadline and release date lie in the interval (that is, $r_j, d_j \in \Delta$), the forward forced demand equals the processing time of the job ($f(j, \Delta) = c_j$). If $c_\tau \leq T_\tau$ for all tasks τ, then each pair of an interval Δ and a task τ can have at most one job j_τ with release date outside the interval ($r_{j_\tau} \notin \Delta$) that has positive forward forced demand ($f(j_\tau, \Delta) > 0$) in the interval.

Definition 3.2. *For a job sequence R of an instance I the* necessary demand $\mathrm{ND}_R(\Delta)$ *of a non-empty interval Δ is the sum of the forward forced demands of all jobs with absolute deadline in Δ. We use* $\mathrm{ND}_R(\Delta, \tau)$ *to denote the part of the necessary demand originating only from jobs of task τ. We write* $\mathrm{ND}(\Delta)$ *and* $\mathrm{ND}(\Delta, \tau)$ *when the sequence R is clear from the context.*

Observe that any algorithm working on any number of speed-1 machines must schedule in an interval at least the necessary demand of that interval.

We use the notation $\mathrm{EDF}_{(m+\mu,\sigma)}$ to denote the scheduling algorithm EDF executed on $(m + \mu)$ speed-σ machines, where ties can be broken arbitrarily.

Definition 3.3. *Consider an instance I and a job sequence R. For a point in time t, a task τ, and a scheduling algorithm A, an interval $\Delta = [t', t)$ is called τ-A-*busy before *t, if executing the algorithm A on the sequence R yields for every point in Δ a positive remaining processing time for at least one of the jobs of task τ.*

Observe that the maximal τ-A-busy interval before t is unique, well defined, and starts with the release date of some job of τ, unless it is empty. Moreover, all demand from τ-jobs released before some maximal τ-A-busy interval Δ is processed by A strictly before Δ.

Theorem 3.4. *Let $\sigma \geq 1$. Consider an instance I which satisfies $c_\tau \leq T_\tau$ and $c_\tau \leq D_\tau$ for all tasks τ. If there is some job sequence R which cannot be scheduled by $EDF_{(m+\mu,\sigma)}$, then there is an interval Δ such that* $\mathrm{ND}_R(\Delta)/|\Delta| > (m + \mu)(\sigma - 1) + 1$.

Before giving the formal, slightly involved proof we convey the main intuitions. Knowing that $\mathrm{EDF}_{(m+\mu,\sigma)}$ fails, we will inductively construct an interval with high load. The interval will be composed of several subintervals. To each subinterval we associate a task such that the subinterval is EDF-busy for that task. Whenever EDF does not process a job of that task in the subinterval, it must have all machines busy. In order to conclude that the load of the whole interval is large, we must establish two things: First, that the fraction of a subinterval, in which its associated task is processed, is small, i.e., in a large part of the subinterval all machines must be busy. Second, everything processed in those busy subintervals is part of the necessary demand of the whole interval.

Proof. From now on we assume that R is a job sequence which cannot be scheduled by $\mathrm{EDF}_{(m+\mu,\sigma)}$, and that t_0 is the first point in time when $\mathrm{EDF}_{(m+\mu,\sigma)}$ fails a deadline.

We define inductively a finite sequence of pairs, comprised of a time t_i and a job j_i, for $1 \leq i \leq z$. For convenience define $\Delta_i := [t_i, t_0)$ and $\overline{\Delta}_i := [t_i, t_{i-1})$. Also the following notation for the work that $\mathrm{EDF}_{(m+\mu,\sigma)}$ does for a job j in a certain measurable subset S of $\mathbb{R}^+$ will be helpful: $\mathrm{EDF}_{(m+\mu,\sigma)}(j, S)$. To shorten we use $m' := (m+\mu)(\sigma-1)+1$.

For each pair (t_i, j_i) we define two subsets of the interval $\overline{\Delta}_i$, namely X_i and Y_i. The first subset X_i is the set of points in time between t_i and t_{i-1} when a job of task τ_{j_i} is processed. Due to the way $\mathrm{EDF}_{(m+\mu,\sigma)}$ schedules, X_i is a finite union of intervals. The other subset is its complement in the interval: $Y_i := \overline{\Delta}_i \setminus X_i$. Further, we set $x_i := |X_i|$ and $y_i := |Y_i|$.

Next, we define two values for each i. They will be interpreted later as certain parts of the work that $\mathrm{EDF}_{(m+\mu,\sigma)}$ does or has to do. Let FAIL be the work that $\mathrm{EDF}_{(m+\mu,\sigma)}$ failed to complete before t_0 for jobs of task τ_{j_1} (so $\mathrm{FAIL} > 0$). We define $\widetilde{W}_i := (m+\mu)\sigma y_i + \sigma x_i$ and $W_i := \sum_{s=1}^{i} \widetilde{W}_s + \mathrm{FAIL}$.

We will show the following properties for our sequence of intervals:

1. $t_0 > t_1 > \ldots > t_z$.

2. During each Y_i all machines are busy.

3. All jobs $\mathrm{EDF}_{(m+\mu,\sigma)}$ schedules during Y_i have a deadline in Δ_i.

4. $W_i > m'|\Delta_i|$.

5. $\mathrm{ND}(\Delta_z) \geq W_z$.

Property 2 implies that $(m+\mu)\sigma y_i = \sum_{j \in J} \mathrm{EDF}_{(m+\mu,\sigma)}(j, Y_i)$ for some set J of jobs.

Basis of the Induction As job j_1 we pick one of the jobs $\mathrm{EDF}_{(m+\mu,\sigma)}$ failed to finish at t_0, though they were due. Among these jobs, the job j_1 is one of those jobs j with largest maximal τ_j-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval before t_0. We let $\overline{\Delta}_1$ $(= \Delta_1)$ be the maximal τ_{j_1}-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval before t_0. This also defines t_1 as the lower endpoint of this interval. Clearly, $t_1 < t_0$ since relative deadlines cannot have zero length.

We have to verify property 2, 3 and 4 for (t_1, j_1). If at a certain time t in the τ_{j_1}-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval $\overline{\Delta}_1$ no job of τ_{j_1} is processed by $\mathrm{EDF}_{(m+\mu,\sigma)}$, then at that time all machines must be busy with jobs that have deadlines not later than t_0. This gives the first two properties. For property 4 we use that $\mathrm{EDF}_{(m+\mu,\sigma)}$ failed at t_0 for j_1:

$$\begin{aligned} W_1 &= \widetilde{W}_1 + \mathrm{FAIL} \\ &> (m+\mu)\sigma y_1 + \sigma x_1 = (m+\mu)\sigma\left(|\Delta_1| - x_1\right) + \sigma x_1. \end{aligned}$$

So we get

$$\frac{W_1}{|\Delta_1|} > (m+\mu)\sigma - (m+\mu-1)\frac{\sigma x_1}{|\Delta_1|}.$$

In Δ_1 the $\mathrm{EDF}_{(m+\mu,\sigma)}$ schedule devotes x_1 units of time on jobs of task τ_{j_1} processing with speed-σ. Since the interval Δ_1 is maximally τ_{j_1}-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy before t_0 and j_1 is not completed within t_0, we know that all those jobs must be released in the interval, and have their deadline in the interval.

The busy interval Δ_1Âăstarts with the release date of some job of task τ_{j_1}. Therefore the number of τ_{j_1}-jobs with release date and deadline in Δ_1 is $\left\lfloor \frac{|\Delta_1| - D_{j_1} + T_{j_1}}{T_{j_1}} \right\rfloor$, and we can bound:

$$\frac{\sigma x_1}{|\Delta_1|} < \frac{c_{j_1}}{|\Delta_1|} \cdot \frac{|\Delta_1| - D_{j_1} + T_{j_1}}{T_{j_1}} \leq \max\left(\frac{c_{j_1}}{D_{j_1}}, \frac{c_{j_1}}{T_{j_1}}\right) \leq 1.$$

To verify the middle inequality one should distinguish the cases $(D_{j_1} \leq T_{j_1})$ and $(D_{j_1} > T_{j_1})$ using $|\Delta_1| \geq D_{j_1}$ for the former.

Combining the two bounds we get property 4:

$$\frac{W_1}{|\Delta_1|} > (m+\mu)(\sigma - 1) + 1 = m'.$$

The Inductive Step Assume that the sequence of pairs up to $i-1$ satisfies the properties. We choose the job j_i as one having the following two properties:

1. The release date of j_i is strictly before t_{i-1}.

2.
$$\mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \bigcup_{s=1}^{i-1} Y_s\right) > f(j_i, \Delta_{i-1}).$$

If no such job can be found, we set $z := i-1$ and return the interval Δ_{i-1} as one justifying the claim of the theorem. We will show later why this holds true. So assume such a j_i exists. Take $\overline{\Delta}_i$ as the maximal τ_{j_i}-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval before t_{i-1}, and accordingly set t_i as its lower endpoint.

Let us show the properties. As the release date of j_i is strictly before t_{i-1}, also $t_i < t_{i-1}$, and we have property 1. The next two properties again follow from the fact that $\overline{\Delta}_i$ is τ_{j_i}-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy. Here, take into account for property 3 that j_i has a deadline in Δ_{i-1} by induction.

To prove property 4 it suffices to show $\widetilde{W}_i \geq m' \left|\overline{\Delta}_i\right|$, because we have strict inequality in $W_{i-1} > m' \left|\Delta_{i-1}\right|$ by induction. By definition

$$\frac{\widetilde{W}_i}{\left|\overline{\Delta}_i\right|} = (m+\mu)\sigma - (m+\mu-1)\frac{\sigma x_i}{\left|\overline{\Delta}_i\right|}.$$

We want to establish $\sigma x_i \leq \left|\overline{\Delta}_i\right|$. Having this, property 4 follows as above.

For this part we simplify notation by setting $\tau := \tau_{j_i}$, $T := T_{\tau_{j_i}}$, $c := c_{\tau_{j_i}}$ and $D := D_{\tau_{j_i}}$. We distinguish the cases $\left|\overline{\Delta}_i\right| \geq T$ and $\left|\overline{\Delta}_i\right| < T$.

Case 1: $\left|\overline{\Delta}_i\right| \geq T$ We can bound σx_i by the amount of work released by τ during the maximal τ-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval $\overline{\Delta}_i$:

$$\sigma x_i \leq \left\lfloor \frac{\left|\overline{\Delta}_i\right|}{T} \right\rfloor \cdot c + \mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \overline{\Delta}_i\right).$$

W.l.o.g. $\left|\overline{\Delta}_i\right|$ is not an integer multiple of T. Otherwise, the last released job could not contribute to the work done in X_i. But then, a slightly smaller value replacing $\left|\overline{\Delta}_i\right|$ would also give a valid bound on what is processed during $\overline{\Delta}_i$.

Recall that $f(j_i, \Delta_{i-1}) := \left(c_{j_i} - (t_{i-1} - r_{j_i})^+\right)^+$. By choice of j_i we know that more than its forced forward demand is done by $\mathrm{EDF}_{(m+\mu,\sigma)}$ in Δ_{i-1}. Therefore

$$\mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \overline{\Delta}_i\right) \leq c - f(j_i, \Delta_{i-1}) \leq (t_{i-1} - r_{j_i}) \leq \left|\overline{\Delta}_i\right| - T \cdot \left\lfloor \frac{\left|\overline{\Delta}_i\right|}{T} \right\rfloor.$$

Note that the middle inequality is also true for $f(j_i, \Delta_{i-1}) = 0$. To verify the last inequality, assume first that j_i is the last job of task τ released in $\overline{\Delta}_i$.

Then between the release of j_i and the end of $\overline{\Delta}_i$ at most $\left|\overline{\Delta}_i\right| - T \cdot \left\lfloor \frac{\left|\overline{\Delta}_i\right|}{T} \right\rfloor$ units of time may pass.

Now, say j_i is not the last job of task τ released in $\overline{\Delta}_i$. Remember that j_i is not finished by $\mathrm{EDF}_{(m+\mu,\sigma)}$ within $\overline{\Delta}_i$. Therefore all jobs of τ released later are not processed within $\overline{\Delta}_i$ at all, because EDF implies FIFO for the jobs of a common task. If there is such a job released but not started in $\overline{\Delta}_i$, we can subtract its entire processing time from the upper bound on σx_i. This means to subtract at least as much as when we subtract $\mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \overline{\Delta}_i\right)$. Thus, we have

$$\sigma x_i \leq \left\lfloor \frac{\left|\overline{\Delta}_i\right|}{T} \right\rfloor \cdot c \leq \frac{\left|\overline{\Delta}_i\right|}{T} \cdot c \leq \left|\overline{\Delta}_i\right| .$$

To finish the case $\left(\left|\overline{\Delta}_i\right| \geq T\right)$ plug everything together:

$$\frac{\sigma x_i}{\left|\overline{\Delta}_i\right|} \leq \frac{\left\lfloor \frac{\left|\overline{\Delta}_i\right|}{T} \right\rfloor \cdot c + \left|\overline{\Delta}_i\right| - \left\lfloor \frac{\left|\overline{\Delta}_i\right|}{T} \right\rfloor \cdot T}{\left|\overline{\Delta}_i\right|} = 1 - \frac{(T-c)\left\lfloor \frac{\left|\overline{\Delta}_i\right|}{T} \right\rfloor}{\left|\overline{\Delta}_i\right|}.$$

As $T \geq c$ we have $\sigma x_i \leq \left|\overline{\Delta}_i\right|$.

Case 2: $\left|\overline{\Delta}_i\right| < T$ Assume $\left|\overline{\Delta}_i\right| < T$. Then only one job of task τ can be released during $\left|\overline{\Delta}_i\right|$, namely j_i. The choice of j_i gives

$$c = \sigma x_i + \mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \Delta_{i-1}\right) \geq \sigma x_i + \mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \bigcup_{s=1}^{i-1} Y_s\right) > \sigma x_i + f(j_i, \Delta_{i-1}).$$

As the release date of j_i is in $\overline{\Delta}_i$ we can use $t_{i-1} - r_{j_i} \leq \left|\overline{\Delta}_i\right|$ (indeed we have equality here) to conclude that

$$c > \sigma x_i + f(j_i, \Delta_{i-1}) = \sigma x_i + c - (t_{i-1} - r_{j_i}) \geq \sigma x_i + c - \left|\overline{\Delta}_i\right| ,$$

which shows $0 > \sigma x_i - \left|\overline{\Delta}_i\right|$ for the case $f(j_i, \Delta_{i-1}) > 0$. Yet, if $f(j_i, \Delta_{i-1}) = 0$ we immediately have $\left|\overline{\Delta}_i\right| \geq c \geq \sigma x_i$.

So we again obtain $\sigma x_i \leq \left|\overline{\Delta}_i\right|$, yielding property 4.

The Breaking Condition. In each step from $i-1$ to i the interval is strictly extended backwards to the release date of at least one job which is released before t_0. As there are finitely many tasks, and all have positive minimum separation time T, there are finitely many such jobs, and we can make only finitely many steps. So at some point the breaking condition, namely that there is no job j_i with the two required properties, must hold.

If this holds we claim property 5 to be true, i.e., $\mathrm{ND}(\Delta_z) \geq W_z$. In the value W_z we count σx_i for each X_i, because the whole τ-demand processed in a τ-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval is part of the necessary demand of that interval. Also, the demand $\mathrm{EDF}_{(m+\mu,\sigma)}$ failed to process before t_0 is part of the necessary demand of Δ_z. For each Y_i part we count $(m+\mu)\sigma y_i$, which is by property 2 exactly what is processed in those times by $\mathrm{EDF}_{(m+\mu,\sigma)}$. By property 3 all jobs processed in some Y_i have their deadline in the interval Δ_i and therefore also in Δ_z. Finally, there is no job among those processed in some section Y_i with release date before t_z, which has been counted in the term $(m+\mu)\sigma y_i$ with more than its forced forward demand in Δ_i. The forced forward demand in the greater interval Δ_z can only be greater, and thus we count for no job more in W_z than in $\mathrm{ND}(\Delta_z)$.

$\square$

We required $c_\tau \leq T_\tau$ and $c_\tau \leq D_\tau$. Both are easy to test in linear time. In fact, the later condition is necessary for scheduling any job sequence on any number of machines with speed 1. The first condition is necessary for scheduling all job sequences on any number of speed-1 machines.

Now, consider $\sigma \geq 2 - \frac{1+\mu}{m+\mu}$. We get $m' = (m+\mu)(\sigma-1)+1 \geq m$. Then, if an instance I allows for a job sequence R with an interval Δ generating a necessary demand $\mathrm{ND}_R(\Delta) > m|\Delta|$ as in the theorem, then clearly it cannot be scheduled by any algorithm on m speed-1 machines. So, all three conditions of the theorem, $c_\tau \leq T_\tau$, $c_\tau \leq D_\tau$, and $\mathrm{ND}_R(\Delta) \leq m|\Delta|$, are necessary for scheduling on m speed-1 machines. By the theorem they are sufficient for scheduling on $(m+\mu)$ speed-σ machines. Therefore, all that is missing for an approximate feasibility test is a procedure testing whether an instance I can have a job sequence R with an interval Δ generating a necessary demand $\mathrm{ND}_R(\Delta) > m|\Delta|$. For this we will provide an FPTAS in the remainder. As this procedure determines the maximal load only up to an ϵ, we will have to choose σ slightly bigger than $2 - \frac{1+\mu}{m+\mu}$ in our final theorem.

3.5 An FPTAS for Load Estimation

The following observation facilitates the test:

Lemma 3.5. *Assume $c_\tau \leq T_\tau$ and $c_\tau \leq D_\tau$ for all tasks τ of an instance I. Then, over all intervals $\Delta = [t, t+\ell)$ of a fixed length ℓ and all job sequences R of I, the maximal necessary demand from a certain task τ is*

$$\mathrm{ND}_{R^*}(\Delta^*, \tau) = c_\tau k + [c_\tau + \ell - D_\tau - kT_\tau]^+, \text{ where } k = \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor .$$

Proof. Rewrite $c_\tau k + [c_\tau + \ell - D_\tau - kT_\tau]^+ = c_\tau k + [c_\tau - (T_\tau - (\ell - D_\tau - (k-1)T_\tau))]^+$. Make $t^* + \ell$ the deadline of some job j from τ and $t^* \geq T_\tau$. Further choose R^* such that all jobs of task τ released in $[t - T_\tau, t + \ell)$ precede their follower at the minimum distance T_τ. Then the necessary demand $\mathrm{ND}_{R^*}(\Delta^*, \tau)$ is as claimed.

To see that this is maximal, assume any interval Δ with $|\Delta| = \ell$ and any job sequence R of I with higher necessary demand than the one in the above construction. As $c_\tau \leq T_\tau$, at most $k+1$ jobs can contribute to $\mathrm{ND}_R(\Delta, \tau)$. Compressing the distances between all contributing jobs cannot diminish the forced forward demand in the interval for any of those jobs. Now push the compressed sequence of contributing jobs towards the right until the deadline of the last job coincides with the right boundary of Δ. This will not diminish the forced forward demand of any contributing job. Thus, we arrive at a job sequence and an interval as in the above construction which generate at least as much forced forward demand as the pair (R, Δ) with which we started. This contradicts that $\mathrm{ND}_R(\Delta, \tau) > c_\tau k + [c_\tau + \ell - D_\tau - kT_\tau]^+$.

□

The construction of the lemma also shows that the maximal forced forward demand can be achieved for each task independently. As a consequence we only have to find the optimal length of an interval. Then we know how much forced forward demand a maximal pair of interval and job sequence has. We define for any instance I satisfying for all $\tau : c_\tau \leq T_\tau$ and $c_\tau \leq D_\tau$:

$$w := w_I : \mathbb{R}^+ \to \mathbb{R}^+, \ell \mapsto w(\ell) := w_I(\ell) := \sum_{\tau \in I} c_\tau k + [c_\tau + \ell - D_\tau - kT_\tau]^+$$

Lemma 3.5 states that $w_I(\ell)$ is the maximum forced forward demand of any job sequence of I in any interval of length ℓ.

The following algorithm finds a length ℓ' which approximates the maximum of $\frac{w(\ell)}{\ell}$ by a factor of ϵ in time polynomial in the input size of I and $1/\epsilon$. In fact, we devise a function ϕ which pointwise approximates the load, i.e.,

$\forall \ell \in \mathbb{R}^+ : (1-\epsilon)\frac{w(\ell)}{\ell} \leq \phi(\ell) \leq \frac{w(\ell)}{\ell}$. There is a polynomial size subset of $\mathbb{R}^+$, a priori determinable, in which the function ϕ must achieve its maximum. So, the approximation algorithm is straightforward.

```
Algorithm 1: Load Estimation($I, \epsilon$)
    For each $\tau \in I$, compute:

    $\text{threshold}(\tau) := D_\tau + T_\tau/\epsilon,$
        $\text{points}(\tau) := \{\ell \in (0, \text{threshold}(\tau)] : \ell = q \cdot T_\tau + D_\tau \text{ for some } q \in \mathbb{N}\},$
        $\text{points}'(\tau) := \{\ell \in (0, \text{threshold}(\tau)] : \ell = q \cdot T_\tau + D_\tau - c_\tau \text{ for some } q \in \mathbb{N}\}.$

    Compute $\text{POINTS} := \cup_{\tau \in I} (\text{points}(\tau) \cup \text{points}'(\tau) \cup \{\text{threshold}(\tau)\})$.
    Output $\lambda := \max\left(\max_{\ell \in \text{POINTS}} \frac{w(\ell)}{\ell}, \sum_{\tau=1}^{n} \frac{c_\tau}{T_\tau}\right)$.
```

Lemma 3.6. *For any instance I Algorithm 1 outputs a λ such that $(1-\epsilon)\lambda^* \leq \lambda \leq \lambda^*$ where $\lambda^* = \sup_{\Delta,R} \frac{\text{ND}_R(\Delta)}{|\Delta|}$, and has running time polynomial in n and $1/\epsilon$.*

Proof. We know that $\lambda^* = \sup_\ell \frac{w(\ell)}{\ell}$. We show that for all $\ell \geq 0$ the function

$$\phi(\ell) := \sum_{\tau:\text{threshold}(\tau)\geq\ell} \frac{w_\tau(\ell)}{\ell} + \sum_{\tau:\text{threshold}(\tau)<\ell} \left(1 - \frac{D_\tau}{\ell}\right)\frac{c_\tau}{T_\tau}$$

approximates the load $w(\ell)/\ell$ in the following sense:

$$(1-\epsilon)\frac{w(\ell)}{\ell} \leq \phi(\ell) \leq \frac{w(\ell)}{\ell}.$$

Secondly, we will show that we can find the maximum of ϕ by only considering points in POINTS. The number of points in POINTS is obviously polynomial in the input, and so is the evaluation of ϕ for each point. This completes the proof.

Recall that

$$w_\tau(\ell) = c_\tau \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor + \left[c_\tau + \ell - D_\tau - \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor \cdot T_\tau\right]^+.$$

Therefore $w_\tau(\ell)T_\tau \geq c_\tau(\ell - D_\tau)$ and

$$\frac{w_\tau(\ell)}{\ell} \geq \frac{c_\tau}{T_\tau} \cdot \left(1 - \frac{D_\tau}{\ell}\right),$$

which summed over all tasks τ yields the upper bound on ϕ.

Concerning the lower bound, for $\ell >$ threshold(τ) we have $\ell > D_\tau + \frac{T_\tau}{\epsilon}$ implying $\epsilon > \frac{T_\tau}{\ell - D_\tau}$. Using again $w_\tau(\ell)T_\tau \geq c_\tau(l - D_\tau)$ gives $\epsilon > \frac{c_\tau}{w_\tau(\ell)}$.

As the difference between the necessary demand of one task τ and the approximate demand $(\ell - D_\tau) \cdot \frac{c_\tau}{T_\tau}$ can at most be the execution time of the task, c_τ, we can substitute

$$\frac{w_\tau(\ell) - (\ell - D_\tau) \cdot \frac{c_\tau}{T_\tau}}{w_\tau(\ell)} < \epsilon$$

and by rewriting we get

$$\frac{\ell - D_\tau}{\ell} \cdot \frac{c_\tau}{T_\tau} > (1 - \epsilon)\frac{w_\tau(\ell)}{\ell} .$$

Again, summing over all tasks gives the claimed lower bound on ϕ.

To finish, observe that between two consecutive points $\ell_1, \ell_2 \in$ POINTS$\cup \{0\}$ we can write

$$\phi(\ell) = C_1/\ell + C_2 + \xi(\ell), \quad \forall\, \ell \in [\ell_1, \ell_2),$$

with

$$\begin{aligned}
C_1 &:= \sum_{\tau:\text{threshold}(\tau)\geq\ell_1} w_\tau(\ell_1) - \sum_{\tau:\text{threshold}(\tau)<\ell_1} \frac{c_\tau D_\tau}{T_\tau} \\
C_2 &:= \sum_{\tau:\text{threshold}(\tau)<\ell_1} \frac{c_\tau}{T_\tau} \\
\xi(\ell) &:= (1/\ell) \cdot \sum_{\tau:\text{threshold}(\tau)\geq\ell_1} \left([c_\tau + \ell - D_\tau - k_1 T_\tau]^+ - [c_\tau + \ell_1 - D_\tau - k_1 T_\tau]^+\right)
\end{aligned}$$

where $k_1 := \left\lfloor \frac{\ell_1 + T_\tau - D_\tau}{T_\tau} \right\rfloor$.

By definition of POINTS, the function ξ can be written as $C/\ell + C'$ for some constants C, C'; this implies that the same is true for the function ϕ inside each interval $[\ell_1, \ell_2)$. Thus, a maximum of ϕ is always attained at an extreme point of such an interval. Also, beyond the maximum of POINTS, the function ϕ equals $\sum_{\tau\in I}\left(1 - \frac{D_\tau}{\ell}\right) \cdot \frac{c_\tau}{T_\tau}$. Therefore, the overall maximum of ϕ is attained at one of the points in POINTS or equals $\sum_{\tau\in I}\frac{c_\tau}{T_\tau}$, and the algorithm is correct.

□

Theorem 3.7. *There exists a feasibility test that, given a task system I, $\mu \in \mathbb{N}$ and $\epsilon > 0$, decides whether I can be scheduled by EDF on $(m + \mu)$ speed-$(1 + \frac{m}{(m+\mu)(1-\epsilon)} - \frac{1}{m+\mu})$ machines, or I cannot be scheduled at all on m speed-1 machines. The running time is polynomial in n, m and $1/\epsilon$.*

Proof. With the help of Algorithm 1 we can verify in polynomial time the following conditions:

(C1) For all tasks $\tau \in I : c_\tau \le \min(D_\tau, T_\tau)$.

(C2) There is $\lambda \le m$, where $(1-\epsilon)\lambda^* \le \lambda \le \lambda^*$ and $\lambda^* = \sup_{R,\Delta} \frac{\mathrm{ND}_R(\Delta)}{|\Delta|}$.

Both are necessary for scheduling I on m speed-1 machines.

Condition (C2) implies that there is no job sequence R and interval Δ such $\mathrm{ND}_R(\Delta) > \frac{m}{1-\epsilon}|\Delta|$. Choosing $\sigma \ge (1 + \frac{m}{(m+\mu)(1-\epsilon)} - \frac{1}{m+\mu})$ gives $(m+\mu)(\sigma-1)+1 \ge \frac{m}{(1-\epsilon)}$, and the claim follows from Theorem 3.4. $\square$

Corollary 3.8. *There exists a feasibility test that, given a task system I and $\epsilon > 0$, decides whether I can be scheduled by EDF on m speed-$(2-1/m+\epsilon)$ machines, or I cannot be scheduled at all on m speed-1 machines. Its running time is polynomial in n, m and $1/\epsilon$.*

4 RECOVERABLE ROBUSTNESS

The pivotal idea of this chapter is well represented by the object depicted in Figure 4.1. This type of object is used in the airline industry. The one shown in the picture was attached at the airport of Sao Paulo to a suitcase of a passenger flying to Munich via Paris Charles de Gaulle. The scheduled time for transit at Charles de Gaulle airport was two hours. Before the time of boarding in Sao Paulo it became apparent that the aircraft would start late. Taking into account the time buffer for the flight time across the Atlantic, the resulting new transfer time of the passenger in Paris could be estimated to seventy-five minutes. This amount of time is not sufficient for a transfer at Charles de Gaulle. Transferring the luggage onto the plane to Munich already takes longer. The connection was highly likely to be lost. Therefore, the ground staff handling the luggage at Sao Paulo attached that flag to the suitcase. Upon arrival in Paris it triggered a priorized handling of the suitcase, accelerating its way through the airport such that the transfer became possible again. A simple piece of paper provided for the recovery of the passenger's original flight connection.

To honor the truth: it did not work. The passenger was deferred on a plane to Munich two hours later. Still, the case displays an idea both natural and successful for coping with uncertainty in practice of operations research. The ingredients of this idea are the following.

In the *planning phase* a schedule is constructed in pursuit of some objective. In the example, it lies in the passenger's interest to have a short connection in order to minimize his total travel time. This is mirrored in the transfer times of the connections offered to the passenger.

But the planning must also take into account that the given *data is subject*

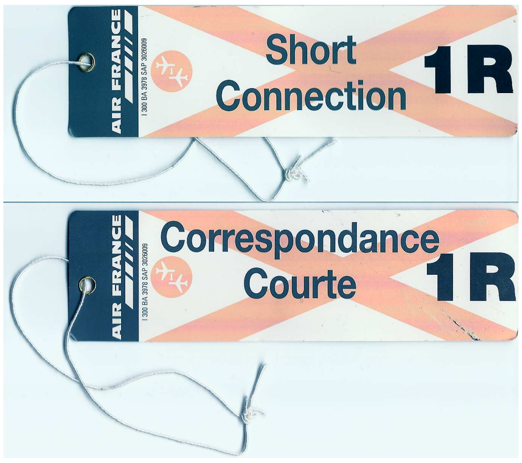


Figure 4.1: Simple means of recovery.

to uncertainty. Delays are not unusual in public transportation systems. Therefore, a plan already incorporates a certain buffer time. In this case, the delay upon arrival in Paris was smaller than that at departure in Sao Paulo, due to buffer time for the flight across the Atlantic. Plans are not to be constructed exceedingly tight.

Despite the buffering, in some scenarios the plan cannot be operated in a straight forward way. These are not necessarily situations of exceptional disruptions, like a total failure of the aircraft, or bankruptcy of the carrier, or natural catastrophes. Even ordinary disruptions may exhaust and even exceed the buffers. Despite the plans not being tight they cannot be constructed so loose that they suit for *all* likely scenarios. Otherwise, the system looses too much of its performance. *Plans must be recovered.*

The *conditions for recovering* a plan differ substantially from the original planning situation. As an advantage for the recovery the exact data, at least of the past events, is known. But the downside of this information is large: Constructing the original flight plan, the carrier uses an elaborate system of optimization and other planning procedures, which can be time consuming and require significant computational resources. Such resources are usually not at the disposal of the planner during recovery. Moreover, a flight connection like the one from Sao Paulo to Munich is planned as part of a huge system. In the event of recovery only a small part of that system can be adjusted. For example, the schedule of the ground crews at Charles de Gaulle is planned after the legs have been planned. But, when the leg from Sao Paulo is delayed, the freedom for re-planning the ground crews is very limited. The recovery must be local and accomplished by simple means of computation and implementation. Such a simple implementation of a recovery is the piece of paper shown in Figure 4.1.

In short we face the following situation in practice:

- Planning consists of an optimization problem.
- Crucial data is notoriously subject to limited changes during operations.
- During operations the solutions are recovered by simple and limited means.

This structure can be observed in several examples from practice. In many of these examples the means of recovery are currently designed in an unsystematic way. Is it possible to construct the plan and the strategy for recovery *jointly*, as a symbiotic pair? Thereby, the constructed plan shall be specifically helpful for the envisioned recovery. And the chosen recovery strategy shall rely on resources that can be provided for free or at low cost in the planning phase. Can such an integrated approach of planning and recovery lead to a more reliable, or a more efficiently operating system? Are problems of this type practically tractable for reasonably sized real-world instances? What is the right model for this type of integrated optimization?

To summarize our goal: Can we find a cheapest plan that is recoverable by restricted means in all likely scenario? We will provide an affirmative answer to this question by means of mathematical optimization. To this end, we introduce the notion of *recoverable robustness.*

The above motivation starts from an informal and not systematically optimized practice, and asks for a concept that allows to enhance this practice by means of optimization. This direction is complemented by considerations starting from an existing mathematical concept for optimization under imperfect information, namely, *robust optimization.* Applying this classical concept to construct delay resistant railway timetables a central weakness of the standard notion of robustness becomes apparent. The solutions found by the classical concept, which we call strict robustness, turn out to be necessarily over-conservative. We will outline briefly the concept of strict robustness and the weakness which stimulated the development of recoverable robustness. (A detailed and technical account of this matter is deferred to a point when the formal definitions are given.) In this way we motivate the new notion both from a purely practical observation and the evolution of existing mathematical concepts.

Overview of the Chapter This chapter is organized in five sections. In the first section we define recoverable robustness in its full scope. In Section 4.2 we consider recoverable robustness for railway timetabling and platforming (a subsequent step to timetabling). This section aims at some understanding of the virtues and limits of the new notion in practice. In the remainder we concentrate on recoverable robustness in the context of linear programming. First, we take a detailed look on some related work (Section 4.3) that is particularly inspiring for the our line of thought. The last two sections contain most of the mathematical content of this chapter. In Section 4.4 we show how to achieve recoverable robustness for recovery by linear programming and right-hand side uncertainty. This includes in particular the robust network buffering problem. The final Section 4.5 takes a polyhedral perspective to recoverable robustness by linear programs. This yields the basis for analyzing the so-called coincidental covering of scenarios by recovery robust solutions, and a polynomial approach to solve recovery robust problems for matrices with vertically and horizontally limited disturbances.

4.1 The Notion of Recoverable Robustness

The general goal of optimization under imperfect information is to find best solutions although full and reliable data is not available. This way the goal is not formulated precisely. Consider an every day example for a decision under imperfect information: Shall one take an umbrella when leaving the house? There are two scenarios. Either it will rain, or it will not rain. How shall we assess, whether it is *better* to take the umbrella or not? It may be clear, in case we know for sure, which of the scenarios is going to take place. But without this information the rationales for deciding the umbrella question can differ significantly. This is not due to the complicated mathematical structure of the umbrella question, but due to the unclarity of what 'best solution' means in a situation of imperfect information. Not surprisingly, there are many concepts for optimization under imperfect information. One might adopt a worst case or an average case approach, or try to optimize an expected value or some risk measure. The question, which approach the real-world decision shall be based on, cannot be answered mathematically. Still, a thorough, mathematical understanding of what the different concepts offer is a prerequisite for any responsible, non-mathematical decision on the matter.

There is no silver bullet for optimization under imperfect information. Therefore, it is important to understand our new concept as an evolution of other concepts, driven by their short comings for particular applications. The new concept, which we present in this chapter, is a heir of a well established tradition, and we will point out repeatedly its relation to other concepts for optimization under imperfect information. Recoverable robustness is a generalization of the classical concept of robust optimization. Robust optimization is an alternative to stochastic programming, mostly to 2-stage stochastic programming.

Note that the term 'robust' is not used consistently in the literature. The notion of recoverable robustness is situated in the tradition of robust optimization originating from Soyster [54]. This field of research can be seen as a special branch of stochastic programming. There is a special issue [4] of the mathematical optimizations journal for the field of robust optimization.

The full notion of recoverable robustness, which we will present, has a very broad scope. One specialization of it extends the concepts of classical robust optimization and overcomes its inherent problem of over-conservatism.

Robustness and 2-Stage Programming The fundamental idea of *robustness* is to construct a solution that is *feasible in all* (likely) *scenarios.*

In contrast, a *2-stage stochastic program* features a scenario independent first stage decision, and for each scenario a second stage decision, which is taken after the full, precise data is known. Together the first and the second stage decision must form a feasible solution in the scenario. The second stage decision usually comes at a higher cost. For example, in case of unexpected heavy rain one can buy an umbrella on the fly. This is more expensive, than carrying the old one. Still, if rain is unlikely, it may be an overall good strategy not to take the umbrella in the first place, and to rely on buying in case of exceptional rain. The 2-stage stochastic program optimizes a mixed objective, summing the deterministic first stage cost and the expected value of the second stage cost. Sometimes the expectation is replaced by a more sophisticated stochastic function including some risk measure. In all cases the scenario set is assumed to be endowed with a probability distribution.

An alternative model optimizes under the restriction that a certain part of the constraints of the optimization problem, usually a linear program, is fulfilled with a given (high) probability. This is called *chance constraint programming*. In general, 2-stage stochastic programming is a special case of *multi-stage stochastic programming*. Here the data is revealed in several steps, after each of which further decisions are taken. The area of stochastic programming is a vast field of research, to which we can by no means do justice in this work. The interested reader is deferred to an extensive literature of textbooks [2, 1, 3] and the excellent web-site of the community [56].

For 2-stage stochastic programs it is generally required that distributions are given in some form and that they can be handled in the solution procedure. Against this background the virtues of the robust approach become visible:

- Robust solutions require no knowledge about distributions.
- Robust models are usually easier to solve, because the solver need not handle distributions, nor look explicitly at each scenario to assess the objective.
- Facing uncertain data robust solutions have a guarantee of quality that is not subject to uncertainty.

The last point is particularly important. In some applications expected values are simply not the right goal. Nobody wants to use an airplane that flies save on average. In general, the expected value is more suitable, if one is interested in the average performance of a solution in several realizations.

While high tractability enables the use of robust methods for large scale instances, and the specific guarantee connected to a robust solution makes them the model of choice for many applications, the downside of robust solutions is also apparent. It is a bit like sweat shirts in the eighties: One-size-fits-all yields a rather conservative outfit in the end. In the umbrella example, robust optimizers would always have to carry the umbrella, unless sunshine is sure. Therefore, restricting the scenario set to *likely* scenarios is very important for robust optimization. We will discuss this in detail in 4.3.2. Smart truncations of the scenario set help to avoid over-conservatism.

Still, there are applications for which those truncations lack any effect. Among these applications is delay resistant railway timetabling. The disturbances affect the driving and stopping times of the trains. Imagine a timetable that is feasible for all elements of a (truncated) scenario set. No matter how the set is truncated, the robust timetable must for each driving or stopping activity respect the maximal duration over all scenarios. This one-fits-all plan will be unacceptably conservative. Railway timetables must be constructed such that they are recovered slightly in case of disturbances. This is a kind of second stage decision. Not taking the possibility of recovery into account yields over-conservative, i.e., exceedingly expensive solutions. We will present a way to plan with respect to this second stage decision without taking an explicit look at the huge number of scenarios that arise in these applications. This way we will find solutions with acceptable costs, without sacrificing the virtues of robust optimization: The compact models and the deterministic guarantee of feasibility under uncertain data.

Again, the different approaches to optimization under imperfect information are suitable for different situations. While a stochastic program is well suited for few but significantly different scenarios, a simple robust model can be adequate for example if the data is subject to small errors of measurement. For the case of delay resistant timetabling stochastic programming is in conflict with the size of typical instances, and the standard robust approach is necessarily over-conservative. Therefore, it requires a new approach to optimization under imperfect information.

4.1.1 Basic Definitions

We are looking for solutions to an optimization problem which in a limited set of scenarios can be made feasible, or *recovered*, by a limited effort. Therefore, we need to define

- the **o**riginal optimization problem (Step O),
- the imperfection of information, that is the **s**cenarios (Step S), and
- the limited **r**ecovery possibilities (Step R).

For Step O and Step S a large toolbox for modeling can be borrowed from classical approaches to optimization respectively optimization with imperfect information. Step R is a little less obvious, and we choose to formalize it via a *class $\mathcal{A}$ of admissible recovery algorithms*.

A solution x for the optimization problem defined in Step O is *recovery-robust*

- *against* the imperfection of information (Step S) and
- *for* the recovery possibilities (Step R),

if in all situations that may occur according to Step S, we can recover from x a feasible solution by means of one of the algorithms given in Step R.

Computations in recovery-robust optimization naturally decompose into a *planning phase* and a *recovery phase*. In the planning phase,

- we compute a solution x which may become infeasible in the realized scenario,
- and we choose $A \in \mathcal{A}$, i.e., one of the admissible recovery algorithms.

Such a pair (x, A) hedges for data uncertainty in the sense that in the recovery phase

- algorithm A is used to turn x into a feasible solution in the realized scenario.

The output (x, A) of the planning phase is more than a solution, it is a *precaution*. It does not only state that recovery is possible for x, but explicitly specifies *how* this recovery can be found, namely by the algorithm A.

The formal definition of recoverable robustness [43] we give next is very broad. The theorems in this chapter will only apply to strong specializations of that concept.

We prepare some terminology. Let $\mathcal{F}$ denote the original optimization problem. An instance $O = (P, f)$ of $\mathcal{F}$ consists of a set P of feasible solutions, and an objective function $f : P \to \mathbb{R}$ which is to be minimized.

By $\mathcal{R} = \mathcal{R}_{\mathcal{F}}$ we denote a model of imperfect information for $\mathcal{F}$ in the sense that for every instance O we specify a set $S = S_O \in \mathcal{R}_{\mathcal{F}}$ of possible scenarios. Let P_s denote the set of feasible solutions in scenario $s \in S$.

We denote by $\mathcal{A}$ a class of algorithms called *admissible recovery algorithms*. A recovery algorithm $A \in \mathcal{A}$ solves the *recovery problem*, which is a feasibility problem. Its input is $x \in P$ and $s \in S$. In case of a *feasible recovery*, $A(x, s) \in P_s$.

Definition 4.1. *The triple* $(\mathcal{F}, \mathcal{R}, \mathcal{A})$ *is called a* recovery robust optimization problem, *abbreviated RROP. A pair* $(x, A) \in P \times \mathcal{A}$ *consisting of a* planning solution x *and an admissible algorithm* A *is called a* precaution.

A precaution is recovery robust, *iff for every scenario* $s \in S$ *the recovery algorithm* A *finds a feasible solution to the recovery problem, i.e., for all* $s \in S$ *we have* $A(x, s) \in P_s$.

An optimal precaution *is a recovery robust precaution* (x, A) *for which* $f(x)$ *is minimal.*

Thus, we can quite compactly write an RROP instance as

$$\boxed{\begin{array}{c} \inf\limits_{(x,A)\in P\times\mathcal{A}} f(x) \\ \text{s.t.} \quad \forall s \in S : A(x, s) \in P_s \quad . \end{array}}$$

The objective function value of an RROP is infinity, if no recovery is possible for some scenario with the algorithms given in the class $\mathcal{A}$ of admissible recovery algorithms.

It is a distinguished feature of this notion that the planning solution is explicitly accompanied by the recovery algorithm. In some specializations the choice of the algorithm is self-understood. For example, for linear recovery robust programs, to which we will devote our main attention, the algorithm is some solver of a linear program or a simpler algorithm that solves the specific type of linear program that arises as the recovery problem of the specific RROP. Then we will simply speak of the planning solution x, tacitly combining it with the obvious algorithm to form a precaution.

4.1.2 Restricting the Recovery Algorithms

The class of admissible recovery algorithms serves as a very broad wildcard for different modeling intentions. Here we summarize some important types of restrictions that can be expressed by means of that class.

The definition of the algorithm class $\mathcal{A}$ also determines the computational balance between the planning and the recovery phase. For all practical purposes, one must impose sensible limits on the recovery algorithms (otherwise, the entire original optimization problem could be solved in the recovery phase, when the realized scenario is known). In very bold term, these limits fall into two categories:

- limits on the actions of recovery;
- limits on the computational power to find those actions of recovery.

We mention two important subclasses of the first category:

Strict Robustness We can forbid recovery entirely by letting $\mathcal{A}$ consist of the single recovery algorithm A with $A(x, s) = x$ for all $s \in S$. This is called *strict robustness*. Note that by strict robustness the classical notion of robust programming is contained in the definition of recoverable robustness.

Recovery Close to Planning An important type of restrictions for the class of admissible recovery algorithms is that the recovery solution $A(x, s)$ must not deviate too far from the original solution x according to some measure of distance defined for the specific problem. For some distance measures one can define subsets $P_{s,x} \subseteq P_s$ depending on the scenario s and the original solution x, such that the restriction to the recovery algorithm that $A(x, s)$ will not deviate too far from x, can be expressed equivalently by requiring $A(x, s) \in P_{s,x}$. As an example, think of a railway timetable that must be recovered, such that the difference between the actual and the planned arrival times is not too big, i.e., that the delay is limited.

4.1.3 Passing Information to the Recovery

If (as it ought to be) the recovery algorithms in $\mathcal{A}$ are allowed substantially less computational power than the precaution algorithms in $\mathcal{B}$, we may want to pass some additional information $z \in Z$ (for some set Z) about the instance to the recovery algorithm. That is, we may compute an extended precaution $B(P, f, S) = (x, A, z)$, and in the recovery phase we require $A(x, s, z) \in P_s$.

As a simple example, consider a class of admissible recovery algorithms $\mathcal{A}$ that is restricted to computational effort linear in the size of a certain finite set of weights, which is part of the input of the RROP instance. Then it might be helpful to pass an ordered list of those weights on to the recovery algorithm, because the recovery algorithm will not have the means to calculate the ordered list itself, but could make use of it.

In Section 4.2 we present another example, namely rule based delay management policies, which shows that it is a perfectly natural idea to preprocess some values depending on the instance, with which the recovery algorithm becomes a very simple procedure.

4.1.4 Limited Recovery Cost

The recovery algorithm A solves a feasibility problem, and we did not consider any cost incurred by the recovery so far. There are at least two ways to do so in the framework of recoverable robustness. Let $d(y^s)$ be some (possibly vector valued) function measuring the cost of recovery $y^s := A(x, s)$.

- **Fixed Limit:** Impose a fixed limit λ to $d(y^s)$ for all scenarios s.
- **Planned Limit:** Let λ be a (vector of) variable(s) and part of the planning solution. Require $\lambda \geq d(y^s)$ for every scenario s, and let $\lambda \in \Lambda$ influence the objective function by some function $g : \Lambda \to \mathbb{R}$.

In the second setting, the planned limit λ to the cost of recovery is a variable chosen in the planning phase and then passed to the recovery algorithm A. It is the task of A to respect the constraint $\lambda \geq d(y)$, and it is the task of the planning phase to choose (x, A, λ), such that A will find a recovery for x with cost less or equal to λ. Therefore, and to be consistent with previous notation we formulate the cost bound slightly different. Let P'_s denote the set of feasible recoveries for scenario s. Then we define P_s by:

$$A(x, s, \lambda) \in P_s :\Leftrightarrow d(A(x, s)) \leq \lambda \wedge A(x, s) \in P'_s$$

We obtain the following recovery robust optimization problem with recovery cost:

$$\boxed{\begin{array}{c} \min\limits_{(x,A,\lambda) \in P \times \mathcal{A} \times \Lambda} f(x) + g(\lambda) \\ \text{s.t.} \quad \forall s \in S : A(x, s, \lambda) \in P_s \quad . \end{array}}$$

Including the possibility to pass some extra information $y \in Y$ to A we obtain:

$$\boxed{\begin{array}{c} \min\limits_{(x,A,z,\lambda) \in P \times \mathcal{A} \times Z \times \Lambda} f(x) + g(\lambda) \\ \text{s.t.} \quad \forall s \in S : A(x, s, z, \lambda) \in P_s \quad . \end{array}}$$

These recovery cost aware variants allow for computing an optimal tradeoff between higher flexibility for recovery by a looser upper bound on the recovery cost, against higher cost in the planning phase. This is conceptually close to two-stage stochastic programming, however, we do not calculate an expectation of the second stage cost, but adjust a common upper bound on the recovery cost. This type of problem still has a purely deterministic objective. The linear recovery robust programs discussed later are an example of this type of RROP.

4.2 Examples of RROPs

4.2.1 Sporadic Real-time Scheduling

The problem we consider in Chapter 3 fits into the framework of recoverable robustness. Given the specification of a sporadic real-time task system, the goal is to design a platform x of (identical) processors (specifying their speed and number) together with a scheduling algorithm A. This pair (x, A) is a precaution. The scheduling algorithm must respect strong computational restrictions. It must be an element of a prescribed family of admissible recovery algorithms. Finally, for every scenario, i.e., for every job sequence respecting the minimal separation time, the recovery algorithm must produce a feasible schedule on the chosen processor platform x. The platform shall be cheap, or small, or energy-efficient, i.e., we are given a cost function $c(x)$, which we want to minimize. This forms an RROP.

4.2.2 Recovery Robust Timetabling

Punctual trains are probably the first thing a layman will expect from robustness in railways. Reliable technology and well trained staff highly contribute to increased punctuality. Nevertheless, also modern railway systems feature small disturbances in every-day operations.

A typical example for a disturbance is a prolongated stop at a station because of a jammed door. A disturbance is a seminal event in the sense that the disturbance may cause several delays in the system but is not itself caused by other delays. Informing passengers about the reason for a delay affecting them, railway service providers sometimes do not distinguish between disturbances, i.e., seminal events, and delays that are themselves consequences of some initial disturbance. We will use the term *disturbance* exclusively for initial changes of planning data. A *delay* is any difference between the planned point in time for an event and the time the event actually takes place. We also speak of *negative delay*, when an event takes place earlier than planned.

A good timetable is furnished with *buffers* to absorb small disturbances, such that they do not affect the planned arrival times at all, or that they cause only few delays in the whole system. Those buffer times come at the expense of longer, planned travel times. Hence they must not be introduced excessively. Delay resistant timetabling is about increasing the planned travel times as little as possible, while guaranteeing the consequences of small disturbances to be limited.

We will now show how delay resistant timetabling can be formulated as a recovery robust optimization problem. We actually show that a robust version of timetabling is only reasonable, if it is understood as a recovery robust optimization problem. Moreover, we show how recoverable robustness integrates timetabling and the so-called delay management. Delay management is the term coined for the set of operational decisions reacting to concrete disturbances, i.e., the recovery actions. Its integration with timetabling is an important step forward for delay resistant timetabling, which can be formalized by the notion of recoverable robustness.

Step O The original problem is the deterministic timetabling problem. It exists in many versions that differ in the level of modeling detail, the objective function, or whether periodic or aperiodic plans are desired. The virtues of recovery robust timetables can already be shown for a simple version.

A Simple Timetabling Problem The input for our version of timetabling is a directed graph $G = (V, E)$ together with a non-negative function $t : E \to \mathbb{R}_+$ on the arc set. The nodes of the graph $V = V_{\text{AR}} \cup V_{\text{DP}}$ model arrival events (V_{AR}) and departure events (V_{DP}) of trains at stations. The arc set can be partitioned into three sets representing driving of trains from one station to the next, E_{DR}, stopping of a train at a station, E_{ST}, and transfers of passengers from one train to another at the same station, E_{TF}. For driving arcs $e = (i, j) \in E_{\text{DR}}$ we have $i \in V_{\text{DP}}$ and $j \in V_{\text{AR}}$, for the two other types $e = (i, j) \in E_{\text{ST}} \cup E_{\text{TF}}$ the contrary holds: $i \in V_{\text{AR}}$ and $j \in V_{\text{DP}}$. The function $t(e)$ expresses the minimum time required for the action corresponding to $e = (i, j)$, in other words the minimum time between event i and event j. For example, for a driving arc e the value of the function $t(e)$ expresses the technical driving time between the two stations.

A feasible timetable is a non-negative vector $\pi \in \mathbb{R}_+^{|V|}$ such that $t(e) \leq \pi_j - \pi_i$ for all $e = (i, j) \in E$. W.l.o.g. we can assume that G is acyclic.

For the objective function we are given a non-negative weight function $w : E \to \mathbb{R}_+$, where $w_e = w(e)$ states how many passengers travel along arc e, i.e., are in the train during the execution of that action, or change trains according to that transfer arc. An optimal timetable is a feasible timetable that minimizes the total planned (or *nominal*) travel time of the passengers:

$$\sum_{e=(i,j)\in E} w_e(\pi_j - \pi_i).$$

Thus the original problem is a linear program:

$$\begin{aligned} \min & \sum_{e=(i,j)\in E} w_a(\pi_j - \pi_i) \\ \text{s.t.} \quad & \pi_j - \pi_i \geq t(e) \quad \forall e = (i, j) \in E \end{aligned}$$

$$\pi \geq 0$$

Step S We assume uncertainty in the time needed for driving and stopping. Those actions typically produce small disturbances. For a scenario s we are given a function $t^s : E \to \mathbb{R}_+$, with the properties $t^s(e) \geq t(e)$ for all $e \in E$, and $t^s(e) = t(e)$ for all $e \in E_{\mathrm{TF}}$. As we only want to consider scenarios with small disturbances, we restrict to those scenarios where $t^s(e) - t(e) \leq \Delta_e$, for some small, scenario independent constant Δ_e. Note that by scaling the rows in the linear program we can set w.l.o.g. $\Delta_e = \Delta$ for all $e \in E$. Additionally, we require that not too many disturbances occur at the same time, i.e., in every scenario for all but k arcs $e \in E$ we have $t^s(e) = t(e)$.

Of course, there are situations in practice where larger disturbances occur. But it is not reasonable to prepare for such catastrophic events in the published timetable.

Strict Robustness The above restrictions to the scenario set can be very strong, in particular, if we choose $k = 1$. But even for such a strongly limited scenario set strict robustness leads to unacceptably conservative timetables. Namely, the strict robust problem can be formulated as the following linear program:

$$\min \sum_{e=(i,j)\in E} w_e(\pi_j - \pi_i)$$

$$\text{s.t.} \quad \pi_j - \pi_i \geq t(e) + \Delta \quad \forall e = (i,j) \in E$$

$$\pi \geq 0$$

In other words, even if we assume that in every scenario at most one arc takes Δ time units longer, we have to construct a timetable as if all (driving and stopping) arcs were Δ time units longer. This phenomenon yields solutions so conservative that classical robust programming is ruled out for timetabling. Indeed, delay resistant timetabling has so far been addressed by stochastic programming [57, 44] only. These approaches suffer from strong limitations to the size of solvable problems.

The real world expectation towards delay resistant timetables includes that the timetable can be adjusted slightly during its operation. But a strict robust program looks for timetables that can be operated unchanged despite disturbances. This makes the plans too conservative even for very restricted scenario sets. Robust timetabling is naturally *recovery robust timetabling* as we defined it. Naturally, a railway timetable has to be robust against *small* disturbances and for *limited* recovery.

Step R The recovery of a timetable is called *delay management.* The two central means of delay management are delaying events and canceling transfers. Delaying an event means to propagate the delay through the network. Canceling a transfer means to inhibit this propagation at the expense of some passengers loosing their connection.

Pure delay propagation seems not deserve the name recovery at all. But recall that if delay propagation is not captured in the model, as in the strict robust model, the solutions become necessarily over-conservative. Delay is a recovery, and though it is a basic, it is a very important.

Actually, delay management has several other possibilities for recovery. For example, one may cancel train trips, re-route the trains, or re-route the passengers by advising them to use an alternative connection, or hope that they will figure such a possibility themselves. Moreover, delay management has to pay respect to several other aspects of the transportation system. For example, the shifts of the on-board crews are affected by delays. These in turn may be subject to subtile regulations by law or contracts and general terms of employment.

We initially adopt a quite simple perspective to delay management gradually increasing the complexity of the model. First we concentrate on delay, later on delay and broken transfers, and finally we add re-routing of passengers. The latter will be used in model that is shown to be PSPACE-hard.

Simple Recovery Robust Timetabling First, we describe a model where the recovery can only delay the events but cannot cancel transfers. This seems not a recovery in the ordinary understanding of the word. It is simply the propagation of delay. But this simple recovery already rids us from the conservatism trap of strict robustness.

In the recovery phase, when the scenario s and its actual driving and stopping times t^s are known, we construct a *disposition timetable* $\pi^s \in \mathbb{R}_+^{|V|}$ fulfilling the following feasibility condition:

- The disposition timetable π^s of scenario s must be feasible for t^s, i.e.,

  $$\forall e = (i, j) \in E : \pi_j^s - \pi_i^s \geq t^s(e).$$

  These inequalities define the set (actually, the polytope) P_s of feasible recoveries in scenario s.

If this was the complete set of restrictions to the recovery, every timetable would be recoverable. We set up limits to the recovery algorithms:

TTC The disposition timetable is bounded by the original timetable in a very strict manner: Trains must not depart earlier than scheduled, i.e.,

$$\forall e \in E_{\mathrm{DP}} : \pi^s(e) \geq \pi(e).$$

This is what we call the *timetabling condition.*

L1 We want the sum of the delays of all arrival events to be limited. Therefore assume we are also given a weight function $\ell : V_{\mathrm{AR}} \to \mathbb{R}_+$ that states how many passengers reach their final destination by the arrival event i. We fix a limit $\lambda_1 \geq 0$ and require:

$$\sum_{i \in V_{\mathrm{AR}}} \ell(i)\,(\pi_i^s - \pi_i) \leq \lambda_1.$$

L2 One may additionally want to limit the delay for each arrival separately, ensuring that no passenger will experience an extreme delay exceeding some fixed $\lambda_2 \geq 0$, i.e.:

$$\forall i \in V_{\mathrm{AR}} : \pi_i^s - \pi_i \leq \lambda_2.$$

In our model a recovery algorithm $A \in \mathcal{A}$ must respect all three limits. The bounds λ_1 and λ_2 can be fixed a priori, or made part of the objective function. In this way upper bounds on the *recovery cost* can be incorporated into the optimization process. For a timetabling problem (G, t, w) and a function $\ell : V_{\mathrm{AR}} \to \mathbb{R}_+$ and constants $g_1, g_2 \geq 0$ and an integer k we can describe the first timetabling RROP by the following linear program:

$$\min \sum_{e=(i,j) \in E} w_e(\pi_j - \pi_i) + g_1 \cdot \lambda_1 + g_2 \cdot \lambda_2$$

$$\begin{aligned}
\text{s.t.} \quad \pi_j - \pi_i &\geq t(e) && \forall e = (i,j) \in E && (4.1)\\
\pi_j^s - \pi_i^s &\geq t^s(e) && \forall s \in S, \forall e = (i,j) \in E && (4.2)\\
\pi_i^s &\geq \pi_i && \forall s \in S, \forall i \in V_{\mathrm{DP}} && (4.3)\\
\sum_{i \in V_{\mathrm{AR}}} \ell(i)\,(\pi_i^s - \pi_i) &\leq \lambda_1 && \forall s \in S && (4.4)\\
\pi_i^s - \pi_i &\leq \lambda_2 && \forall s \in S, \forall i \in V_{\mathrm{AR}} && (4.5)
\end{aligned}$$

$$\lambda_{\{1,2\}}, \pi^s, \pi \geq 0$$

The set of scenarios S in this description is defined via the set of all functions $t^s : E \to \mathbb{R}_+$ which fulfill the following four conditions from Step S:

$$t^s(e) \geq t(e) \quad \forall e \in E$$

$$t^s(e) \leq t(e) + \Delta \quad \forall e \in E$$
$$t^s(e) = t(e) \quad \forall e \in E_{\mathrm{TF}}$$
$$|\{e \in E : t^s(e) \neq t(e)\}| \leq k$$

In our terminology Inequality (4.1) defines P, Inequality (4.2) defines P_s, Inequalities (4.3) to (4.5) express limits to the action of the algorithm, namely, that the recovery may not deviate to much from the original solution. In detail (4.3) models the TTC, (4.4) ensures condition L1 and (4.5) condition L2.

Here and in the remainder of the example we use mathematical programs to *express* concisely the problems under consideration. These programs are not necessarily the right approach to *solve* the problems.

Breaking Connections In practice delay management allows for a second kind of recovery. It is possible to cancel transfers in order to stop the propagation of delay through the network. We now include the possibility to cancel transfers into the recovery of our model.

Again we restrict to a simple version for explanatory purposes. A transfer arc e can be removed from the graph G at a fixed cost $g_3 \geq 0$ multiplied with the weight w_e. With a sufficiently large constant M we obtain a mixed integer linear program representing this model:

$$\min \sum_{e=(i,j)\in E} w_e(\pi_j - \pi_i) + g_1 \cdot \lambda_1 + g_2 \cdot \lambda_2 + g_3 \cdot \lambda_3$$

$$\begin{aligned}
\text{s.t.} \quad \pi_j - \pi_i &\geq t(e) && \forall e = (i,j) \in E && (4.6)\\
\pi_j^s - \pi_i^s &\geq t^s(e) && \forall s \in S, \forall e = (i,j) \in E_{\mathrm{DR}} \cup E_{\mathrm{ST}} && (4.7)\\
\pi_j^s - \pi_i^s + Mx_e^s &\geq t^s(e) && \forall s \in S, \forall e = (i,j) \in E_{\mathrm{TF}} && (4.8)\\
\pi_i^s &\geq \pi_i && \forall s \in S, \forall i \in V_{\mathrm{DP}} && (4.9)\\
\sum_{i\in V_{\mathrm{AR}}} \ell(i)\,(\pi_i^s - \pi_i) &\leq \lambda_1 && \forall s \in S && (4.10)\\
\pi_i^s - \pi_i &\leq \lambda_2 && \forall s \in S, \forall i \in V_{\mathrm{AR}} && (4.11)\\
\sum_{e\in E_{\mathrm{TF}}} w_e x_e^s &\leq \lambda_3 && \forall s \in S && (4.12)
\end{aligned}$$

$$\lambda_{\{1,2,3\}}, \pi^s, \pi \geq 0$$
$$x^s \in \{0,1\}^{|E_{\mathrm{TF}}|}$$

In our terminology Inequality 4.6 defines P. Inequalities 4.7 and 4.8 define P_s for every s. Again Inequalities 4.9 to 4.11 express limits to the actions that can be taken by the recovery algorithms. These are limits to the deviation of the recovered solution π^s from the original solution π.

4.2.3 Computationally Limited Recovery Algorithms

So far we imposed limits on the *actions* of the recovery algorithms. But delay management is a real-time task. Decisions must be taken in very short time. Thus it makes sense to impose further restrictions on the computational power of the recovery algorithm. Note that in general such restrictions cannot be expressed by a mathematical program as above. We now give two examples for computationally restricted classes of recovery algorithms.

The Online Character of Delay Management In fact the above model has a fundamental weakness. It assumes that the recovered solution, i.e., the disposition timetable $\pi^s = A(\pi, s)$ can be chosen after s is known completely. This is of course not the case for real-world delay management: The disturbances evolve over time, and delay management must take decisions before the whole scenario is known. This means that the algorithms in $\mathcal{A}$ must be *non-anticipative*.

To give a formal definition of non-anticipative algorithms, let $\tau_{\pi,s,A}(e)$ be the point in time when the travel or stopping time $t^s(e)$ becomes known, in case delay management is done by algorithm A for a timetable π. Note that the time $\tau_{\pi,s,A}(e)$ may indeed depend on the timetable π, the scenario s, and on the delay management A. Further let π^{s_1} and π^{s_2} be two disposition timetables computed by the same recovery algorithm A for the same solution π and scenarios s_1 and s_2. We define $E_{\pi,s,A,\tau} = \{e \in E | \tau_{\pi,s,A}(e) \leq \tau\}$ to be the set of all arc times $t^s(e)$ known at time τ. A recovery algorithm A is *non-anticipative* if

$$\forall \tau_0 \in \{\tau \in \mathbb{R}_+ | E_{\pi,s_1,A,\tau} = E_{\pi,s_2,A,\tau} \wedge t^{s_1}(e) = t^{s_2}(e) \, \forall e \in E_{\pi,s_1,A,\tau}\}$$

we have

$$\pi^{s_1}(e) \leq \tau_0 \Rightarrow \pi^{s_1}(e) = \pi^{s_2}(e)$$

and, vice versa,

$$\pi^{s_2}(e) \leq \tau_0 \Rightarrow \pi^{s_2}(e) = \pi^{s_1}(e).$$

In prose, being non-anticipative means to handle scenarios alike, up to the time when they show a first difference. Note that as formalized here the notion will not suit randomized algorithms. It is conceivable how this could be extended. We will not discuss randomized algorithms (although they might be helpful for delay management). It is a natural, computational restriction for delay management to be non-anticipative.

PSPACE-hardness of Delay Management The multistage structure of some delay management models, namely that uncertain events and dispatching decisions alternate, makes these problems extraordinarily hard. We present a model that is still quite restricted but already PSPACE-hard.

The complexity class PSPACE contains those decision problems, which can be decided with the use of memory space limited by a polynomial in the input size. The class NP is contained in PSPACE, because in polynomial time only polynomial space can be used. It is widely assumed that NP is a proper subset of PSPACE. Given this, one cannot decide in polynomial time that a given solution to a PSPACE-hard problem is feasible, because else the solution would be a certificate and therefore the problem in NP. (Note that the complexity terminology is formulated for decision problems. Feasibility in this context means that the delay management solution is feasible in the usual sense and in addition has cost less or equal to some constant.) The difficulty to assess the quality of a solution means for delay management that we cannot expect to be able in all cases to compare the quality of two competing strategies.

We now define a delay management problem and sketch a reduction proving it to be PSPACE-hard. We will prove [20] that the following simple version of the online railway delay management problem is already PSPACE-hard, i.e., at least as hard (by polynomial time reduction) as any problem in PSPACE.

For the reduction we use the following PSPACE-complete problem:

Definition 4.2. *Deciding whether a logical expression of the following type is true*

$$\exists x_1 \forall x_2 \ldots \exists x_{n-1} \forall x_n : \bigwedge_i \bigvee_j z_{ij},$$

where z_{ij} are literals in the variables $\{x_1, \ldots, x_n\}$ and their negations, is called the Quantified Boolean Formula (QBF) *problem.*

We will reduce QBF to a simple version of the online delay management problem.

The Basic Online Delay Management Problem. An instance of the *basic online delay management* (BODM) problem

$$(G, T, C, \pi, S, \mathcal{D})$$

is a six-tuple comprising an infrastructure graph G, a set of trains T, a directed graph C with a vertex set $V(C)$ of relevant events and an arc set $A(C)$ representing precedence constraints among those events, a timetable

$\pi : V(C) \to \mathbb{R}_{\geq 0}$, a set S containing functions $\tau : A(C) \to \mathbb{R}_{\geq 0}$ (which we call *scenarios*) for the minimal time distances of two events connected by a precedence constraint, and finally some mathematical object $\mathcal{D}$ expressing a cost model for the delay management.

The interpretation of the infrastructure graph may be adapted to the specific problem. Usually, a station is represented by a vertex and tracks are represented by edges. In a different interpretation a station can be modelled in more detail an therefore consist of several vertices and edges.

The vertex set of C, is the set of relevant events. Each relevant event is characterized by a triple (t, a, b), where $t \in T$ is a train and $a, b \in V(G) \cup E(G)$ are either vertices or edges of the infrastructure graph. The triple (t, a, b) represents the event that train t changes from infrastructure vertex (edge) a to infrastructure edge (vertex) b. The timetable entry $\pi((t, a, b))$ states the time for which this event is scheduled.

Disposition Timetable The goal is to give a non-anticipative strategy that constructs a feasible *disposition timetable* in every scenario. A disposition timetable is a vector $\pi' : V(C) \to \mathbb{R}_{\geq 0}$ that respects the timetabling condition $\pi' \geq \pi$. It is feasible in a scenario τ, if for all precedence constraints $a = (i, j) \in A(C)$ we have $\pi(j) - \pi(i) \geq \tau(a)$.

As the strategy is required to be non-anticipative, the data $\tau((x, y))$ is revealed only after the time when $\pi'(x)$ took place.

Cost Model Different ways to define the cost model $\mathcal{D}$ are possible. We will use the following: The cost model contains a set of origin-destination pairs with a certain weight, i.e., we know how many passengers want to travel from a certain starting station to a certain final destination. Their paths through the infrastructure network G are fixed. They may follow that path on different trains, but the sequence of stations they pass is fixed. In each scenario the total delay of the passengers in π' compared to π, plus a certain fixed cost for those passengers, who will not reach their destination at all, defines the cost.

An alternative way to define the cost, specifies a certain cost for each transfer that is broken and for each arrival which is delayed in π'. The two models are not tantamount, but can be translated into each other in many cases. For the reduction we will use the model described above. But we will sometimes refer to the cost as the cost of breaking a transfer or delaying a train, because these terms are more convenient, and in the specific case can be translated into the original cost model.

The decision problem, which we show to be PSPACE-hard, is the following question:

Definition 4.3. *Given a BODM instance and a budget B the* BODM decision problem *consists in the following question: Is there a non-anticipative strategy for constructing a disposition timetable, that achieves a cost value lower than the budget B in every realization of $\tau \in S$.*

Theorem 4.4. *The BODM decision problem is PSPACE-hard.*

Reduction of QBF to the BODM Decision Problem For a given Boolean formula in conjunctive normal form, $\bigwedge_i \bigvee_j z_{ij}$, with literals in the set of variables $\{x_1, \ldots, x_n\}$ and their negations we construct an instance of the BODM decision problem. For this BODM instance exists a strategy that achieves a cost lower than the budget B, if and only if the quantified Boolean formula,

$$\exists x_1 \forall x_2 \ldots \exists x_{n-1} \forall x_n : \bigwedge_i \bigvee_j z_{ij},$$

is true. Thus, the BODM decision problem is PSPACE-hard.

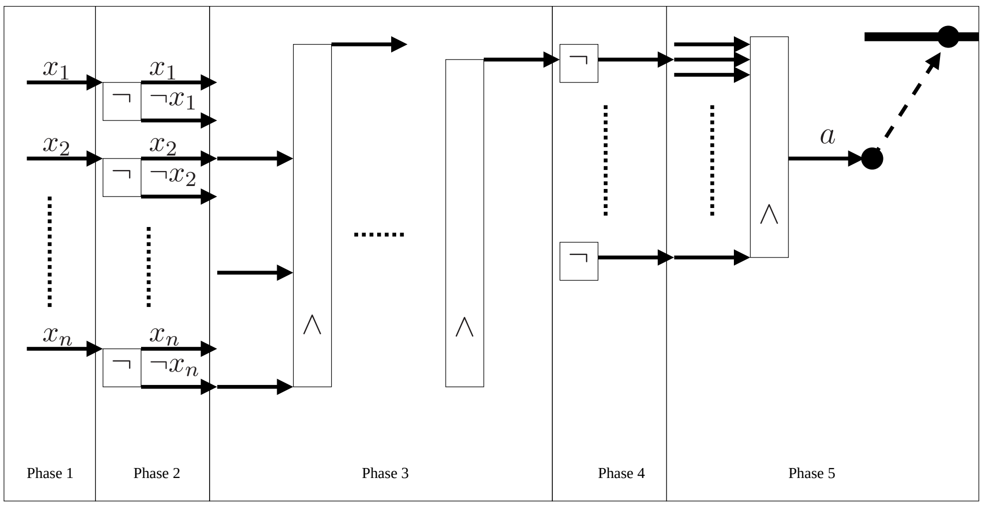


Figure 4.2: Using the Gadgets.

In our construction we use *fixed* and *non-fixed* trains. A train is fixed in the sense that delaying this train would automatically exceed the budget by yielding a cost $M_0 > B$. Nevertheless, we use fixed trains that are a priori fixed to be late. Such a late fixed train has an initial delay prior to the decisions of the strategy, but may neither be delayed any further, nor has a buffer time to compensate the delay. We introduce the late, fixed trains to explain transfers that are a priori broken, i.e., lead from an arrival (of a late, fixed train) to an earlier departure. The cost for the initial delays of those trains is constant for all further unfolding of the scenario and all disposition timetables. Therefore, we can neglect them.

The non-fixed trains fall into two different groups. Each train of the first group, the *variable-trains*, corresponds to a variable x_i of the Boolean formula. If such a train is delayed, we will interpret the corresponding variable x_i as being false, and true if the train is on time. The trains of the second group are called *modeling-trains*, as they serve some technical purpose in the reduction. They can also be delayed or run on time while the strategy is carried out. But the reduction will be constructed such that their delay is entirely dependent on the delay of the variable-trains.

For modeling reasons we want that for every non-fixed train the decision about running delayed or on time must be taken at the start of the train's ride and kept until the final destination. We enforce this by an incoming transfer from a late, fixed train at the beginning of the ride and an outgoing transfer to an on time, fixed train at the end. The non-fixed train cannot meet both transfers. Thus it always incurs the cost for breaking one of these transfers, M_1. Let m be the number of non-fixed trains, then the total budget $M_1 m + C < B < M_1(m + 1) + C$ is set such that none of these trains may break both transfers. (The constant C is the constant cost of all *gadgets*, as explained below.)

With these ingredients (on time fixed trains, late fixed trains, variable-trains, modeling-trains, and the rule that any of the later two types of trains must be scheduled either late in the whole disposition timetable or on time in the whole disposition timetable) we will below device a gadget for a logical NON-operator and a gadget for a logical, multiple AND-operator. The NON-gadget will yield that a certain modeling train is delayed if and only if a certain other train is on time. The AND-gadget has an out-train that is on time, if and only if all trains of a certain set are on time. Before we describe the mechanism of these gadgets, we will first show how they are used to reduce the QBF, $\exists x_1 \forall x_2 \ldots \exists x_{n-1} \forall x_n : \bigwedge_i \bigvee_j z_{ij}$. Actually, we use an alternative way to write the Boolean formula namely, $\bigwedge_i \bigvee_j z_{ij} = \bigwedge_i \neg(\bigwedge_j \neg z_{ij})$.

The time horizon of the constructed BODM instance is split into five phases (cf. Figure4.2).

1. In the first phase all decision whether a variable-train is delayed or not are taken one after the other. The incoming transfer from the fixed train determines the order by which these decisions must be taken. Recall, because of the transfers to the fixed trains these decision cannot be changed later. This way we reflect the consecutive mechanism in the QBF.

   For those variable-trains that correspond to an all-quantified x_i the decision, whether they run late or on time, shall be taken by the adversary. The scenario set S is restricted such that the adversary may produce exactly one of the two following situations: Delay the train i before the incoming transfer from a late, fixed train at the beginning of the ride of train i, or (exclusive or) delay train i immediately before the outgoing transfer to a punctual, fixed train at the end of the ride of train i. Note that the dispatcher cannot intentionally delay a train of an all-quantified variable as the adversary would then have the choice to delay it twice, which would exceed the cost limit.

2. Then each variable-train runs through a NON-gadget, producing its negated train that is late if and only if the variable-train was on time.

3. The variable-trains and their negations pass through the AND-gadgets for each of the re-written clauses. A re-written clause i, $\bigwedge_j \neg z_{ij}$, is modeled by an AND-gadget with in-trains corresponding to the variable-trains or negated variable-trains z_{ij}.

4. Each out-train of those AND-gadgets is negated.

5. Finally all of those negations enter the central AND-gadget. The out-train a of this gadget has a tight transfer to a fixed train. If that out-train is late, it yields a cost of M_2.

We will make sure that every AND- and NON-gadget yields a fixed cost in all scenarios. Thus, we can choose B and M_2 such that the total cost is below B, if the train a is on time, and the cost exceeds B, if a is late. In this way the BODM instance is feasible with cost limit B, if and only if the quantified Boolean formula is true. This completes the reduction.

The NON-Gadget The initial state of a NON-gadget is depicted in Figure 4.3. There is a fixed train (drawn as a bold line) and two non-fixed trains. The lower of the non-fixed trains, the in-train is always late for its transfer to the other non-fixed train. We draw a rhombus to symbolize some fixed delay that should explain this fact. The upper train can wait for the lower train and thus keep the connection (Figure 4.5). But, if the lower train is additionally delayed before the rhombus, the upper train would have to wait so long that is has to break a transfer to a fixed train. The cost for breaking this transfer is M_0, i.e., would immediately exceed the budget. Thus, the strategy will break the transfer from the in-train to the out-train, and the latter will leave the gadget on time (Figure 4.4). A delayed in-train yields an on time out-train, and vice versa.

Still the gadget would not work, because we cannot guarantee that the transfer is not broken, if the in-train is on time, or the out-train is delayed

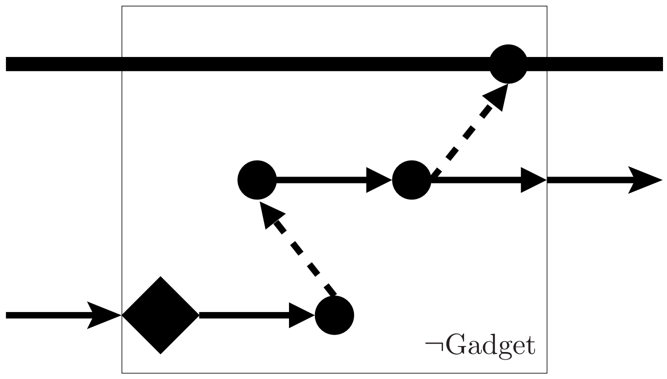


Figure 4.3: The Initial Situation.

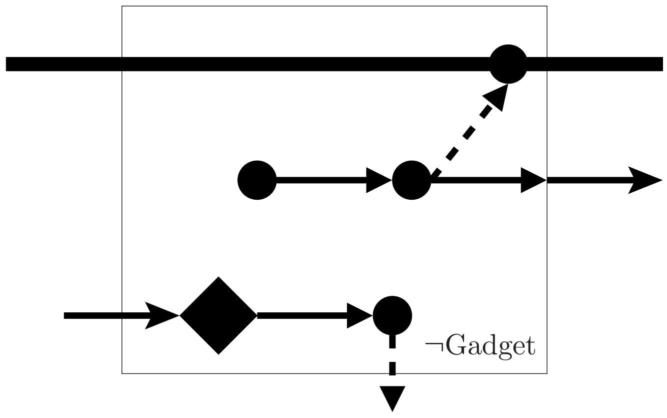


Figure 4.4: Delayed yields On Time.

although it breaks the transfer from a late in-train. To exclude these cases, we have to make sure that a NON-gadget yields a fixed cost in both dispositions we desire (in-train on time & out-train late and vice versa), and exceeds this cost in any other disposition. To this end, let c_w be the cost of delaying the out-train, c_b the cost of breaking the transfer from the in-train, and $c_b - c_w = c_g$ a positive number. We introduce an a priori broken transfer from a late, fixed train to the in-train, which to break costs c_g. This transfer is broken, if and only if the in-train is on time. In other words, the desired dispositions are the only two dispositions by which the gadget has cost less or equal to c_b—and those yield cost equal to c_b.

Note we use NON-gadgets that output the out-train and the in-train and we use NON-gadgets that only output the out-train.

The AND-Gadget The AND-gadget is fairly simple: All in-trains have to be on time for the out-train to be on time. Therefore, all in-trains have

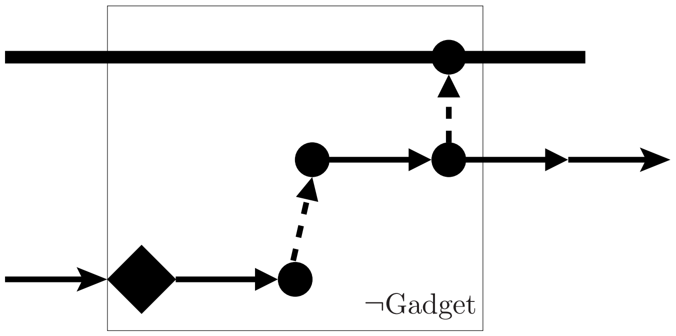


Figure 4.5: On Time yields Delayed.

a tight connection of breaking cost M_0 to the out-train. Again, we have to make sure that the out-train is not scheduled late although all in-trains are on time. To this end, all in-trains run along the same track for some distance. There are some passengers that want to travel this distance, but come from a late, fixed train. Only if at least one of the in-trains is late, these passengers will reach their destination. The cost c_h of not serving these passengers equals the cost of delaying the out-train. Thus, the gadget has at least cost c_h, and will exceed this cost, in case the out-train is late though all in-trains are punctual.

Rule Based Delay Management The previous observation is quite discouraging. How shall one design a recovery robust timetable, if the recovery itself is already PSPACE-hard? We now dicuss a special restriction to the delay management that is motivated by the real-world railway application and turns each decision whether to wait or not to wait into a constant time solvable question. The model keeps the multistage character, but the resulting recovery robust timetabling problem is solvable by a mixed integer program.

Delay management decisions must be taken very quickly. Moreover, as delay management is a very sensitive topic for passengers' satisfaction the transparency of delay management decisions can be very valuable. A passenger might be more willing to accept a decision that is based on explicit rules about how long, e.g., a local train waits for a high-speed train than to accept the outcome of some in-transparent heuristic or optimization procedure. For these two reasons, computational limits for real-time decisions and transparency for the passenger, one may want to restrict the class $\mathcal{A}$ of admissible recovery algorithms to *rule based delay management*. The idea is that trains will wait at most a fixed time for the trains connecting to them, depending on the type of involved trains. For example, a local train might wait 10 minutes for a high-speed train, but vice versa the waiting time could be zero. Fixing these maximal waiting times determines the delay management (within the assumed modeling precision). But, which waiting times are best? Does the asymmetry in the example make sense? We want to optimize the waiting rules, i.e., the delay management together with the timetable.

Assume we distinguish between m types of trains in the system, i.e., we have a mapping $\mu : V \to \{1, \dots, m\}$ of the events onto the train types. A rule based delay management policy A is specified by a matrix $\mathcal{M} = \mathcal{M}_A \in \mathbb{R}_+^{m \times m}$. Its y-th entry in the x-th row m_{xy} is the maximum time a departure event of train type y will be postponed in order to ensure transfer from a type x train. Formally, a rule based delay management policy schedules a departure event j at the earliest time π_j^s satisfying

$$\begin{aligned} \pi_j^s &\geq \pi_i^s + t^s(i,j) && \forall (i,j) \in E_{\mathrm{ST}} \\ \pi_j^s &\geq \min\{\pi_i^s + t^s(i,j), \pi_j + m_{\mu(i)\mu(j)}\} && \forall (i,j) \in E_{\mathrm{TF}} \\ \pi_j^s &\geq \pi_j \,. \end{aligned}$$

Arrival events are scheduled as early as possible respecting TTC and the driving times in scenario s:

$$\pi_j^s = \max(\{\pi_j\} \cup \{\pi_i^s + t^s(i,j) | (i,j) \in E_{\mathrm{DR}}\}) \quad \forall j \in V_{\mathrm{AR}}$$

Actually, the maximum is taken over two elements, as only one driving arc (i,j) leads to each arrival event j.

Moreover, for a transfer arc $(i,j) \in E_{\mathrm{TF}}$ the canceling variable $x_{(i,j)}$ is set to 1 if and only if the result of the above rule gives $\pi_i^s + t(i,j) > \pi_j^s$.

It is easy to see that such a recovery algorithm gives a feasible recovery for every (even non-restricted) scenario s and every solution $\pi \in P$. If we restrict $\mathcal{A}$ to the class of rule based delay management policies, the RROP consists in finding a $m \times m$ matrix $\mathcal{M}$ and a schedule π that minimizes an objective function like those in the models we presented earlier:

$$\min_{\mathcal{M}, \pi, \lambda_{\{1,2,3\}}} \sum_{e=(i,j) \in E} w_e(\pi_j - \pi_i) + g_1 \cdot \lambda_1 + g_2 \cdot \lambda_2 + g_3 \cdot \lambda_3$$

$$\begin{aligned}
\text{s.t.} \quad \pi_j - \pi_i \geq t(e) & \qquad \forall e = (i,j) \in E & (4.13)\\
\forall s \in S, \forall j \in V_{\mathrm{DP}} : & & (4.14)\\
\pi_j^s \;=\; & \max(\{\pi_i^s + t^s(i,j) \mid (i,j) \in E_{\mathrm{ST}}\} & \\
\cup\; & \max\{\min\{\pi_i^s + t^s(i,j), \pi_j + m_{\mu(i)\mu(j)}\} \mid (i,j) \in E_{\mathrm{TF}}\} & \\
\cup\; & \{\pi_j\}) & \\
\forall s \in S, \forall j \in V_{\mathrm{AR}} : & & (4.15)\\
\pi_j^s \;=\; & \max(\{\pi_j\} & \\
\cup\; & \{\pi_i^s + t^s(i,j) \mid (i,j) \in E_{\mathrm{DR}}\} & \\
\pi_j^s - \pi_i^s + M x_e^s \geq t^s(e) & \qquad \forall s \in S, \forall e = (i,j) \in E_{\mathrm{TF}} & (4.16)\\
\sum_{i \in V_{\mathrm{AR}}} \ell(i)\,(\pi_i^s - \pi_i) \leq \lambda_1 & \qquad \forall s \in S & (4.17)\\
\pi_i^s - \pi_i \leq \lambda_2 & \qquad \forall s \in S, \forall i \in V_{\mathrm{AR}} & (4.18)\\
\sum_{e \in E_{\mathrm{TF}}} w_e x_e^s \leq \lambda_3 & \qquad \forall s \in S & (4.19)
\end{aligned}$$

$$\lambda_{\{1,2,3\}}, \pi^s, \pi \geq 0$$
$$x^s \in \{0,1\}^{|E_{\mathrm{TF}}|}$$

The timetabling condition is ensured automatically by the rule based delay management described in Equations (4.14) and (4.15).

Rule based delay management algorithms are non-anticipative. The formulation we give even enforces the following behavior: The departure π_i of a train A will be delayed for transferring passengers from train B (with arrival π_j) for the maximal waiting time $m_{\mu(i)\mu(j)}$, even if before time $\pi_i + m_{\mu(i)\mu(j)}$ it becomes known that train B will arrive too late for its passengers to reach train A at time $\pi_i + m_{\mu(i)\mu(j)}$. As formulated, a train will wait the due time, even if the awaited train is hopelessly delayed. In practice, delay managers might handle such a situation a little less short minded.

Rule based delay management is a good example for the idea of integrating robust planning and simple recovery. Consider the following example of two local trains, A and B, and one high-speed train C. Passengers transfer from A to B, and from B to C. Assume local trains wait 7 minutes for each other, but high-speed trains wait at most 2 minutes for local trains. Then train A being late could force train B to loose its important connection to the high-speed train C. Indeed, this could happen, if the timetable and the waiting times are not attuned. In the planning, we might not be willing to increase the time a high-speed train waits, but instead plan a sufficient buffer for the transfer from B to C. This example demonstrates how waiting times and buffer times in the timetable are intertwined and should be planned together.

4.2.4 Is Recovery Helpful?

So far we gave examples of applications where recovery is key. But there are situations in which recovery is useless. We will now give an example of an application, namely platforming, where prima facie recovery is helpful. But a closer look shows that the line between helpful and useless recovery is very thin for platforming.

Railway planning and optimization proceeds in several subsequent steps. It is a system ranging from network planning to crew assignment and fare systems. Obviously, this cannot be planned in a single step. One classically disconnected step following immediately the timetabling is called *platforming.* A platforming problem considers a single train station. The trains have more or less fixed arrival and departure times at the station, calculated in the timetabling phase. The task of platforming is to assign the trains to the tracks of the station.

Simple Aperiodic Platforming As minimum requirement in platforming each platform is assigned to only one train at a time. Formally, we are given a finite set of intervals $T_i = [a_i, d_i] \subset \mathbb{R}$ of the time axis and an integer k. Interval T_i corresponds to the time train i is planned to stop at the station, and k is the number of available tracks at the station. A conflict free platforming on k platforms corresponds to a feasible coloring of the interval graph $G_{\mathcal{T}}$ arising from the set of intervals $\mathcal{T} := \{T_i\}_i$ with k colors.

An interval graph is a perfect graph, i.e., the chromatic number equals that of the maximal clique. A maximal clique in an interval graph can be found efficiently by scanning along the time axis every point where an interval starts. Thus, the problem can be solved efficiently. We consider the optimization variant where k is to be minimized.

Let us now construct a recovery robust variant of this simple platforming problem: Trains may be delayed or arrive earlier, i.e., the each interval T_i in a scenario can be translated by a real number τ to $T_i + \tau = [a_i + \tau, d_i + \tau]$. We limit these disturbances in the absolute value: $|\tau| \leq \Delta$.

A strictly robust solution has to assign tracks to the trains, respectively colors to the intervals, such that for any combination of translations of intervals no two intervals of the same color contain a common point in time. This problem is equivalent to solving the original problem for a graph, which we call the *strict robust graph* $G_{\mathcal{T}_\Delta}, \mathcal{T}^\Delta := \{[a_i - \Delta, d_i + \Delta] : [a_i, d_i] \in T\}$ instead of $G_{\mathcal{T}}$. This robustification is likely to increase the minimal k substantially.

Replacing strict robustness by recoverable robustness, allows for the following recovery: The platforming can assign to each train a list of up to r different tracks in the planning phase. In a scenario the train will be send to the first, currently free track on its list. In practice, one might want to choose $r = 2$ and add as a further limit for the recovery phase that the two assigned tracks of a train must be at same platform. This ensures that passengers can react to a change of track with a minimal effort.

Can we achieve a smaller k by allowing this kind of recovery? Disregard for a moment the additional requirement of neighboring tracks, i.e., tracks of the same platform. If a recovery without this limitation is of no use, then the further limited recovery also will not help.

Formally, we have the following problem: given the set of interval graphs that can arise as scenarios from $\mathcal{T}$ with disturbances limited by Δ. Assign to each index i a list of r colors, such that in every scenario, we can color the interval graph using for each interval only colors from the list of its index.

At first sight, this relaxes the problem of finding the coloring in the scenario. It will turn out that in fact it makes no difference.

Is there an instance of this problem for which the total number of different colors used in the lists is strictly less than the minimal coloring of $G_{\mathcal{T}^\Delta}$? Unfortunately, the answer is in the negative: Assume the chromatic number of $G_{\mathcal{T}^\Delta}$ is $\bar{k}$. Then there is a point in time t where $\bar{k}$ of the intervals in $\mathcal{T}^\Delta$ coincide. But, then there is also a scenario in which $\bar{k}$ many trains are in the station at time t. Thus, also the recovery robust solution must use $\bar{k}$ different tracks.

In this example the concept of recoverable robustness seems useless. At least the limited recovery used here appears to be unsuitable. But a slight change in the setting can change the picture dramatically.

Simple Periodic Platforming Many transportation systems run a periodic timetable. This means that the different trains of a line arrive (or depart) at a certain station at same time modulo a certain period. For example, a high-speed train leaves every hour at seven minutes past the hour from station A towards station B. Or, consider a line A of a subway system that runs with a period of ten minutes, i.e., if a train of line A departs from a certain station at 11:32h, then the next train of that line will leave the station at 11:42h, another one at 11:52h and so on. Of course, this periodicity should also be reflected in the platforming. The passenger should always find the trains of the same line at the same track.

This can be presented again as a coloring problem on a special graph class, namely, circular-arc graphs. Formally, a circular-arc graph can be defined via a set of intervals $\mathcal{T}$, each representing a line, and the period time P. Two intervals $T_1, T_2 \subset \mathbb{R}$ collide, i.e., are connected by an edge, if

$$\exists x_1 \in T_1, x_2 \in T_2, p \in \mathbb{Z} : x_1 = x_2 + pP.$$

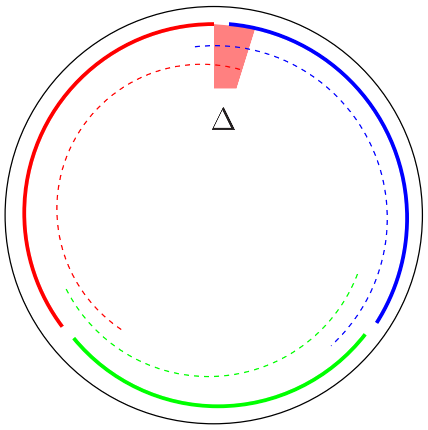


Figure 4.6: The colored arcs represent the periodic stopping times of three lines. The intervals of possible aberration for a small Δ are shown by the dashed lines.

Figure 4.2.4 gives an example with three lines that for any $\Delta > 0$ cannot be platformed strictly robust with less than three tracks, but in the recovery robust setting with $r = 2$ can be platformed with $k = 2$ for any $\Delta < P/6$. The regular stopping times of the three lines fill the entire period P. With only small aberrations from the plan each pair of lines can collide. The dashed lines represent the intervals of the strict robust graph. But, deviation below $P/6$ guarantee that for each point in time only two lines are at the station. The ratio between the minimal number of tracks in strict robust solution and the minimal number in a recovery robust solution is 3/2.

In some situations a limited recovery will not allow to reduce the costs of a robust solution. The examples of simple platforming and simple periodic platforming show that the line between situations where recovery helps and where it does not help can be thin.

4.3 Inspiration from the Tradition of Robust Programming

In the *next* section we present basic results on RROPs, for which the recovery problem is a linear program. In particular, we consider this setting for the case where also the original problem $\mathcal{F}$ is a linear program. We concentrate on the latter case in the presentation, although some of the results hold independently of the type of the original problem. In this section we motivate and illustrate this approach by a comparison to the tradition of robust optimization.

The most similar concept to recoverable robustness is that of adjustable robust counterparts [5]. The adjustable robust counterpart also allows for certain variables of the linear program to be set, after the scenario data is revealed. In [5] the resulting problem is shown to be solvable in polynomial time, if the adjustable variables are linear functions of the random data. Confining recoverable robustness to linear programming, the notions are close enough that one could read the remainder of this chapter as an substantial extension to the analysis of adjustable robust counterparts. But, this is a discussion about names.

4.3.1 Linear Programming Recovery

Consider a linear program ($\min c'x$, s.t. $A^0x \geq b^0$) with m rows and n variables. We seek solutions to this problem that can be recovered by limited means in a certain limited set of disturbance scenarios. The situation in a disturbance scenario s is described by a set of linear inequalities, notably, by a matrix A^s and a right-hand side b^s. We slightly abuse notation when we say that the scenario set S contains a scenario (A^s, b^s), which, strictly speaking, is the image of scenario s under the random variable (A, b). We will discuss later more precisely the scenario sets considered in this analysis. For the linear programming case the limited possibility to recover is defined via a recovery matrix $\hat{A}$, a recovery cost d, and a recovery budget D. A vector x is recovery robust, if for all (A^s, b^s) in the scenario set S exists y such that $A^s x + \hat{A}y \geq b^s$, and $d'y \leq D$. Further, we require that x is feasible for the original problem without recovery, i.e., $A^0x \geq b^0$. The problem then reads:

$$\begin{aligned}
&\inf_x c'x \\
&\quad \text{s.t.} \qquad A^0x \geq b^0 \\
&\qquad \forall (A,b) \in S\ \exists y \in \mathbb{R}^{\hat{n}}: \\
&\qquad\qquad\qquad Ax + \hat{A}y \geq b \\
&\qquad\qquad\qquad d'y \leq D
\end{aligned}$$

When S is a closed set in the vector space $\mathbb{R}^{(m\times n+m)}$ we know that either the infimum is attained, or the problem is unbounded. This case constitutes the principal object of our considerations, the Linear Recovery Robust Program:

Definition 4.5. *Let A^0 be an $m \times n$-matrix called the* nominal matrix*, b^0 be an m-dimensional vector called the* nominal right-hand side*, c be an n-dimensional vector called the* nominal cost vector*, $\hat{A}$ be an $m \times \hat{n}$-matrix called the* recovery matrix*, d be an $\hat{n}$-dimensional vector called the* recovery cost vector*, and D be a non-negative number called the* recovery budget*. Further let S be a closed set of pairs of $m \times n$-matrices and m-dimensional vectors, called the* scenario set*. Then the following optimization problem is called a* Linear Recovery Robust Program (LRP) *over S:*

$$\begin{aligned}
&\min_x c'x \\
&\quad s.t. \qquad A^0x \geq b^0 \\
&\qquad \forall (A,b) \in S\ \exists y \in \mathbb{R}^{\hat{n}}: \\
&\qquad\qquad\qquad Ax + \hat{A}y \geq b \\
&\qquad\qquad\qquad d'y \leq D
\end{aligned}$$

We refer to the A as the *planning matrix* although it is a quantified variable. The planning matrix describes how the planning x influences the feasibility in the scenario. The vectors $y \in \mathbb{R}^{\hat{n}}$ with $d'y \leq D$ are called the *admissible recovery vectors*. Note that we do not call S a scenario *space*, because primarily there is no probability distribution attached to it.

We are unnecessarily restrictive, when requiring the same number of rows for A^0 as for $\hat{A}$ and A. If this is not the case, nothing in what follows is affected, except may be readability.

If a solution x can be recovered by an admissible recovery y in a certain scenario s, we say *x covers s*.

To any LRP we can associate a linear program, which we call the scenario expansion of the LRP:

$$\begin{aligned}
\min_{x,(y^s)_{s\in S}} \quad & c'x \\
\text{s.t.} \quad & A^0x \geq b^0 \\
& A^sx + \hat{A}y^s \geq b^s \quad \forall s \in S \\
& d'y^s \leq D \qquad\quad \forall s \in S
\end{aligned}$$

Note that in this formulation the set S is comprised of the scenarios s, whereas in the original formulation it contains pairs of the form (A^s, b^s). This ambiguity of S is convenient and should cause no confusion to the reader.

The scenario expansion is a first possibility to solve the LRP. But, usually, the scenario set is too big to yield a solvable scenario expansion. The scenario sets which we will consider are not even finite.

We will frequently use an intuitive reformulation of an LRP that can be interpreted as a game of a *planning player* setting x, a *scenario player* choosing (A, b), and a *recovery player* deciding on the variable y. The players act one after the other:

$$\inf_x c'x \text{ s.t. } A^0x \geq b^0 \wedge D \geq \left\{ \sup_{(A,b)\in S} \left\{ \inf_y d'y \text{ s.t. } Ax + \hat{A}y \geq b \right\} \right\} \tag{4.20}$$

with constant vectors $c \in \mathbb{R}^n$, $b^0 \in \mathbb{R}^m$ and $d \in \mathbb{R}^{\hat{n}}$, constant matrices $A^0 \in \mathbb{R}^{m,n}$ and $\hat{A} \in \mathbb{R}^{m,\hat{n}}$, and variables $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m,n}$, $b \in \mathbb{R}^m$ and $y \in \mathbb{R}^{\hat{n}}$.

Again, when it is clear that either the extrema exist or the problem is unbounded we use the following notation:

$$\min_x c'x \text{ s.t. } A^0x \geq b^0 \wedge D \geq \left\{ \max_{(A,b)\in S} \left\{ \min_y d'y \text{ s.t. } Ax + \hat{A}y \geq b \right\} \right\} \tag{4.21}$$

Observe that an LRP, its scenario expansion and its 3-player formulation have the same feasible set of planning solutions x. Whereas, the set of recovery vectors y that may occur as a response to some scenario (A^s, b^s) in the 3-player formulation, is only a subset of the set of feasible second stage solutions y^s in the scenario expansion. The 3-player formulation restricts the later set to those responses y, which are minimal in $d'y$. But this does not affect the feasible set for x.

The formalism of Problem (4.21) can also be used to express that x and y are required to be non-negative. But it is a lot more well arranged, if we state such conditions separately:

$$\min_{x} c'x \text{ s.t. } A^0x \geq b^0 \wedge D \geq \left\{ \max_{(A,b)\in S} \left\{ \min_{y\geq 0} d'y \text{ s.t. } Ax + \hat{A}y \geq b \right\} \right\} \quad (4.22)$$

and

$$\min_{x\geq 0} c'x \text{ s.t. } A^0x \geq b^0 \wedge D \geq \left\{ \max_{(A,b)\in S} \left\{ \min_{y\geq 0} d'y \text{ s.t. } Ax + \hat{A}y \geq b \right\} \right\} \quad (4.23)$$

The purely deterministic condition $A^0x \geq b^0$, which we call *nominal feasibility* condition, could also be expressed implicitly by means of S and $\hat{A}$. But, this would severely obstruct readability. In some applications the nominal feasibility plays an important role. For example, a delay resistant timetable shall be feasible for the nominal data, i.e., it must be possible to operate the published timetable unchanged at least under standard conditions. Else, trains could be scheduled in the published timetable x to depart earlier from a station than they arrive there. However, in this rather technical section the nominal feasibility plays a minor role.

Let us mention some extensions of the model. The original problem may as well be an integer or mixed integer linear program,

$$\min_{x=(\hat{x},\bar{x}),\bar{x}\in\mathbb{Z}} c'x \text{ s.t. } A^0x \geq b^0 \wedge D \geq \left\{ \max_{(A,b)\in S} \left\{ \min_{y} d'y \text{ s.t. } Ax + \hat{A}y \geq b \right\} \right\} \quad (4.24)$$

or some other optimization problem over a set of feasible solutions P and an objective function $c : \mathbb{R}^n \to \mathbb{R}$, in case the *disturbances* are confined to the right-hand side:

$$\inf_{f\in P} c(f) \text{ s.t. } D \geq \left\{ \sup_{b\in S} \left\{ \inf_{y} d'y \text{ s.t. } f + \hat{A}y \geq b \right\} \right\} \quad (4.25)$$

with a fixed planning matrix A.

Using the concept of planned limits to the recovery cost (cf. p. 59), the budget D can also play the role of a variable:

$$\begin{aligned} &\min_{D\geq 0,\, x} c'x + D \text{ s.t. } A^0x \geq b^0 \wedge \\ &D \geq \left\{ \max_{(A,b)\in S} \left\{ \min_y d'y \text{ s.t. } Ax + \hat{A}y \geq b \right\} \right\} \end{aligned} \quad (4.26)$$

In case of right-hand side disturbances only, we can again formulate:

$$\inf_{f \in P, D \geq 0} c(f) + D \text{ s.t. } D \geq \left\{ \sup_{b \in S} \left\{ \inf_{y} d'y \text{ s.t. } f + \hat{A}y \geq b \right\} \right\} \tag{4.27}$$

In the sequel we explain some lines of thought that stem from the tradition of robust optimization and strongly influence the way we shaped and analyze the notion of recoverable robustness. Therefore we take a moderately extensive view in particular on the classical work of Bertsimas and Sim [22].

4.3.2 Truncated Scenario Sets and the Art of Robust Programming

Modern robust optimization, even before the introduction of recoverable robustness, takes a stand in the trade-off between computational tractability and modeling precision for optimization under imperfect information. Strict robustness refrains from considering second stage or recovery actions. This makes calculations easy, but runs the risk of being over-conservative.

Robust solutions would always be over-conservative, if they were constructed with respect to extreme scenarios, too. Modeling a stochastic program one can leave scenarios with extreme disturbances in the model, as long as they can somehow be recovered. The costs for recovery, the second stage costs in an extreme scenario may be high, but their probability is so small that they hardly influence the stochastic solution. In contrast for robust optimization one must not consider extreme scenarios, unless one is ready to take extremely conservative solutions. Therefore, the art of robust optimization is to find truncations to the scenario set that fulfill the following three goals:

1. **(Tractability)** Truncate such that the scenario set yields a tractable, robust optimization model.
2. **(Inexpensiveness)** Truncate such that the scenario set allows reasonably cheap robust solutions.

3a. **(Covering)** Given a probability measure, truncate such that the cut away part has small probability.

The combination of the three goals, namely an easy model that yields reasonably cheap solutions which cover most scenarios, is not a trivial result. Assume, we would like to incorporate into the optimization procedure the following goal: Find the *cheapest* solution x such that $Ax \geq b$ holds for a set of scenarios (A, b) with probability at least 0.9. This poses a so-called chance constrained program. In general such programs have non-convex solution spaces. They are not likely to be tractable. Therefore, a priori finding a smart truncation of the scenario space constitutes an interesting alternative to optimizing over a chance constraint.

One of the pathbreaking ideas in the seminal paper by Bertsimas and Sim ([22]) was to propose the following double truncation of the scenario set for a linear program.

- **(Interval-bound)** First, they assume each entry a^s of the planning matrix to vary only within a certain interval centered at the nominal value a^0. (They assume the right-hand side to be fixed.) Formally: $a^s \in [a^0 - \tilde{a}^s, a^0 + \tilde{a}^s]$.
- **(Γ-bound)** Second, they introduce for each row i separately a parameter Γ_i which bounds the number of entries in that row that differ from the nominal value, i.e., $\forall i : |\{a^s_{ij} \neq a^0_{ij}\}| \leq \Gamma_i$.

The combination of Γ- and interval-bound yields a truncation of the scenario set, which meets the three goals proposed above. The truncation by Γ_i, for a linear program, is easily seen to be equivalent to the following: For each row i, the total normalized deviation from the nominal values is less or equal to Γ_i, i.e., $\sum_j |a^s_{ij} - a^0_{ij}|/\tilde{a}_{ij} \leq \Gamma_i$. This comes in handy to solve the robust problem as a linear program of almost the same size as the original problem. Thus, their truncations meet the first requirement, and the standard model of Bertsimas and Sim can be formulated as follows (we change to maximization problems to be consistent with [22]):

$$\begin{aligned} &\max c'x \\ &\text{s.t.} \quad \forall A \in \left\{(a_{ij})_{(ij)} \in \mathbb{R}^{n\times m} : |a_{ij} - a^0_{ij}| \leq \tilde{a}_{ij} \wedge \textstyle\sum_j \frac{|a_{ij}-a^0_{ij}|}{\tilde{a}_{ij}} \leq \Gamma_i\right\} : \\ &\qquad\qquad Ax \leq b \end{aligned} \tag{4.28}$$

Consider this problem as a game of two players, setting subsequently x and A. The first player wants to maximize $c'x$, but must ensure the inequalities. The second player wants to violate at least one inequality. Equivalently, we can formulate with an objective function for the second player and a third player:

$$\min_x c'x \text{ s.t. } 0 \geq \left\{\max_{(A)\in S}\left\{\min_{y\geq 0} \|y\|_1 \text{ s.t. } Ax + \hat{A}y \geq b\right\}\right\}$$

with an arbitrary recovery matrix $\hat{A}$, and $S = \{(a_{ij})_{(ij)} \in \mathbb{R}^{n\times m} : |a_{ij} - a^0_{ij}| \leq \tilde{a}_{ij} \wedge \sum_j |a_{ij} - a^0_{ij}|/\tilde{a}_{ij} \leq \Gamma_i\}$. The Bertsimas-Sim model (4.28) is a special case of Problem (4.22).

For the second truncation goal, the authors mention explicitly that the parameters Γ_i should be trimmed to trade conservatism for robustness and vice versa. The Γ-bound is a concept to control conservatism. Still, there are examples, like robust timetabling, in which changing the parameters Γ_i is of no help to avoid over-conservatism. These are notably those examples that motivated, what we call a *vertical integration* accomplished by recoverable robustness. We will discuss vertical integration in a separate section. For the time being, observe that the truncation considers the disturbances horizontally (row-wise) integrated: The disturbances of the different entries in the i-th row must *all together* not exceed the Γ_i-bound.

For the third goal, Bertsimas and Sim argue that the truncation by the parameters Γ_i yield solutions, which are feasible with high probability for a quite general class of probability measures on the non-truncated scenario space. Both from the point of view of application and from a stochastic analysis one can support the claim that the solutions robust for this truncation have a high probability to be feasible in the non-truncated space. Note that in the absence of exact knowledge about the distribution of the random data, non-mathematical arguments resting on experience and educated estimation shall not be a priori dismissed. We will start by those practice-based arguments and turn to the mathematical analyis in detail later. One can distinguish three arguments here.

Non-mathematical Arguments for the Truncation Firstly, the use of the parameters Γ_i suits a gut-feeling. The feeling says that not all, and not even many of the data entries are going to change. Most things work as they are supposed to, and only a few fail. For many applications, this makes sense to practitioners. For example, most train rides operate as planned, and only a few cause the trouble to the system. In this way we argue from the application, but tacitly imply a stochastic claim, namely, that situations in which at least one of the two truncations is violated are rare. We will see that this claim can be underpinned mathematically for the second kind of truncation (the Γ bound).

Secondly, the analysis of Bertsimas and Sim shows that a slightly different gut-feeling also has a mathematical support: If we protect against the scenarios in the truncated set, we have a good chance to be feasible as well in other scenarios. This means, even if in modeling we choose the parameters Γ_i so small that the Γ-bound is not highly likely to be fulfilled, the robust solutions found for that Γ-bound are highly likely to be feasible for scenarios outside the Γ-bound. We will study the mathematical arguments for this in depth in Subsection 4.3.4.

Finally, a third argument that intrinsically cannot be derived from a mathematical analysis of the model, appeals from the point of view of several applications. The rationale of truncating the scenario set is that we do not want to design our solution with respect to those extreme scenarios in which either the total deviation from the nominal data, or the local deviation from at least one entry in the nominal data, is beyond a certain threshold. To assess, whether this modeling is justified in light of the application, more factors play a role than only the probability distribution. Even if extreme scenarios occur relatively frequent, say in one out of twenty cases, one might still not be willing to plan the business as usual with respect to those cases. One possible reason is that in extreme scenarios resources and operating possibilities are activated, which are completely uncommon to the used optimization model. In particular, when customer satisfaction is at stake, very high disturbances are accepted by a majority of the customers as a valid excuse for changes and inconvenience in operation. The customers expect the planning to be able to cope perfectly with a limited number of small disturbances. But they show sympathy for the management of a major disturbance causing inconvenience also to them. Though this last argument cannot be justified mathematically, because it explicitly relies on non-modeled features, it seems to us that it constitutes a lot of the attractiveness of such truncations to practitioners.

Stochastic Arguments for the Truncation Unlike the third argument, the first two practical arguments are supported by a stochastic analysis. Bertsimas and Sim show the following. Let the entries of the matrix be a set of random variables, each distributed independently and symmetrically around the mean value, a^0_{ij}. Let x fulfill the robust constraint of the i-th row for some Γ_i. Then the probability that x will not fulfill the constraint in a randomly chosen scenario (not necessarily limited by Γ_i) decreases exponentially with increasing Γ_i, and for fixed Γ_i with increasing n.

Several upper bounds are known on the probability for a sum of independent random variables to deviate from its expected value. These bounds decrease exponentially with the increase of the bound to the deviation, and also hold for weaker stochastic conditions than those required in this model. Consider in particular the inequality of Hoeffding [35]. From these we get immediately that the probability for the sum of normalized deviations to exceed Γ_i decreases exponentially with increasing Γ_i. Bertsimas and Sim use the Markov bound to derive a similar result, too. But, despite the exponential decay, the bound only gives satisfactory results if Γ_i^2 is relatively big, in comparison to n. But the Γ_i should be small to prevent conservatism in the solution. Therefore, they derive a stronger bound, specific for linear programs, that rests on an estimation for what we call *coincidental covering*.

Coincidental covering is an effect typical for robust solutions. A solution that is robust for a certain scenario set, necessarily also covers a certain mass of other scenarios outside this scenario set. The classical stochastic bounds say: choosing relatively small Γ_i means to fix a (in terms of the probability measure) large set of scenarios for which every solution must be feasible. The stronger Bertsimas-Sim bound allows to choose an absolutely (with respect to n) small Γ_i, thus to protect intentionally against a small part of the probability mass. But, the bound shows a priori that solutions protected against those few scenarios coincidentally also cover most other scenarios. The set of scenarios for which feasibility is *required* is small, but the set of scenarios which are *coincidentally covered* is large.

The effect of coincidental covering suggests to reformulate the third truncation goal:

3b. **(Coincidental Covering)** Given a probability space over the scenario set, truncate such that every solution feasible for the truncated scenario set has high probability to be feasible for the whole space.

This phenomenon is a main issue in our analysis of recoverable robustness. Therefore, we recall the corresponding result of Bertsimas and Sim in detail in Subsection 4.3.4.

Before we turn to this one should also remark that the interval-bound, namely, that no matrix entry takes a value outside a certain interval, is a much more audacious claim than the Γ-bound. Even if the probability for each entry to stay in its interval is high—this means that one has chosen large intervals, which will cause conservative results—the probability for all entries to do so, is quite low for large matrices. This is true, because a Bernoulli process of large length (here $m \cdot n$) even with high probability for success, has a considerable probability to have at least one failure. Bertsimas and Sim give no bound for the case that entries have positive (although small) probability to take values outside their interval. For our considerations, we will also assume that each random entry stays inside a certain interval. Though in principal, the structure of *our* analysis for coincidental covering by recovery robust solutions also allows to understand coincidental covering of scenarios, which violate the interval-bound.

To summarize, the truncation of the scenario set is a crucial step in constructing a powerful robust model. The guidelines are the three truncation goals: easy model, cheap solution, and a high probability for the solution to be feasible in general. Considerable mathematical effort has been invested in pursuit of the first and the last goal. Recoverable robustness is designed for so far not systematically addressed, second goal. It creates flexibility that allows for less conservative solutions. Of course, the new concept is more complex, which at first sight causes problems for tractability and a priori stochastic analysis. Nevertheless, we can present positive results both on tractability and on stochastic bounds for recoverable robustness in the remainder. Understanding the Bertsimas-Sim bound in a polyhedral perspective prepares the ground for the rationale of our results on coincidental covering.

4.3.3 Horizontally and Vertically Integrated Protection

Classical robust optimization, 2-stage stochastic programming, and recoverable robustness for linear programs all construct solutions for a set of constraint system, the scenario set. They differ in the way the first stage solution pays respect to the different constraints that can occur in the scenario. Let us compare four models: A 2-stage stochastic linear program with fixed recourse (4.29), a robust linear program (4.30) in the Soyster model [54], a robust linear program (4.31) in the Bertsimas-Sim model [22], and an LRP (4.32).

$$\begin{aligned} \min c'x + & \textstyle\sum_{s\in S} p(s)\cdot d'y^s \\ \text{s.t.} \quad & \forall s \in S: \quad A^s x + \hat{A}y^s \geq b^s \end{aligned} \tag{4.29}$$

$$\begin{aligned} \max\, & c'x \\ \text{s.t.} \quad & Ax + \tilde{A}y \leq b \\ & -y \leq x \leq y \end{aligned} \tag{4.30}$$

$$\begin{aligned} \max\, & c'x \\ \text{s.t.} \quad & \forall A \text{ with } |a_{ij} - a^0_{ij}| \leq \tilde{a}_{ij} \wedge \textstyle\sum_j \frac{|a_{ij}-a^0_{ij}|}{\tilde{a}_{ij}} \leq \Gamma_i : \\ & Ax \leq b \end{aligned} \tag{4.31}$$

$$\begin{aligned} &\min c'x && (4.32)\\ &\text{s.t.} \quad A^0x \geq b^0 \\ &\forall (A,b) \in S\ \exists y \in \mathbb{R}^{\hat{n}} : \\ &\qquad Ax + \hat{A}y \geq b \\ &\qquad d'y \leq D \end{aligned}$$

For the Soyster model (4.30) we use $\tilde{A} = (\tilde{a}_{ij})_{(ij)}$ as the matrix with entry $\tilde{a}_{ij}$ equal to the half length of the interval in which a_{ij} can vary. Note that $-y \leq x \leq y$ implies $y \geq 0$.

The objective of the stochastic program considers each scenario's cost with its probability. Intuitively, the solver may choose a solution that is not suitable for some unlikely scenarios, in the sense that the solution will have high costs in these scenarios. The Soyster model in contrast constructs a solution that suits even the worst value for each entry of the matrix.

As reaction to this conservatism the Bertsimas-Sim model mimics the behavior of a stochastic program by the a priori decision to neglect those scenarios that have more than Γ entries in one row differing from their nominal value. Instead of looking at each entry separately, the Bertsimas-Sim model considers all entries of a row together. They adopt a horizontally integrated perspective.

It is a perfectly natural idea to extend this integration to the vertical direction. The Soyster model assumes that it is unlikely for a single entry of the uncertain matrix to take a value outside a fixed interval. The Bertsimas-Sim model adds that it is unlikely for the disturbances in one row to simultaneously be all very high. By the same token, it is unlikely that the total disturbance in the whole system is beyond a certain threshold.

The problem about this natural extension for the concept of the Γ-bound is that any robust program which cannot model recovery actions, also cannot use any vertically integrated bound. Assume we bound the number of disturbances in a row by $\bar{k}$ and the total number of disturbances by k. Then a robust model without recovery for all $k \geq \bar{k}$ has the same set of feasible solutions as the same model with $k = m\bar{k}$, where m is the number of rows.

In the last model (4.32), the LRP, we can impose such a vertically integrated bound to (A, b) (or b if only right-hand side disturbances occur) and because of the recovery vector y, the solution can benefit from this bound by being less conservative. Note that for right-hand side disturbances the vertical integration is the only useful integration.

4.3.4 The Bertsimas-Sim Bound

In this subsection we will prove a slightly simplified version of Theorem 2 in [22]. The original proof is technical and found in the appendix of [22]. We give a proof of the simplified version that is very elusive for what follows and rests on similar ideas as the original.

One simplification is to assume Γ_i to be integral. This is merely to simplify notation. The second simplification concerns the assumptions on the distributions. In fact, we restrict ourselves to the worst of the distributions allowed in the original theorem. Thus, any proof that the simplified theorem considers the worst distribution, would make the simplified theorem equivalent to the original. Formally, the original theorem provides for this proof. It is not in our interest to re-prove the bound but to drag its roots to the light.

Return to Problem (4.28). Assume we are given a feasible solution x to this robust optimization problem. The Bertsimas-Sim bound states that the feasibility of the solution will prevail with a considerable probability, when the adversary's choice is replaced by any random variable A, which is fully symmetrically distributed over $\bigotimes_{i,j}[a^0_{ij} - \tilde{a}_{ij}, a^0_{ij} + \tilde{a}_{ij}]$. What is meant by fully symmetric, is explained in the following definition.

Definition 4.6. *For each $1 \leq i \leq q$ let $Z_i = (\Omega_i, \mathcal{F}_i, \mu_i)$, $\Omega_i \subseteq \mathbb{R}$, be a probability space such that for every measurable set $S \in \mathcal{F}_i$ we have $-S := \{x | \exists y \in S : x = -y\} \in \mathcal{F}_i$, and*

$$\mu_i(z_i \in S) = \mu_i(z_i \in -S)$$

holds. Define $Z = \bigotimes_{i=1}^n Z_i$ as the product space and denote an element of Ω by $z = (z_1, \dots z_n)$. We say such a vector z is fully symmetrically distributed over *$\bigotimes_{i=1}^q \Omega_i$.*

Note that for a fully symmetrically distributed vector z we have independence of the coordinates, i.e., $\mu(z \in \bigotimes_i S_i) = \prod_i \mu(z : z_i \in S_i)$, for any set of $S_i \in \mathcal{F}_i$.

A solution to Problem (4.28) is protected against the Γ-limited worst-case. We want to prove that this solution is likely to stay feasible without the Γ-bound, if the disturbance will oscillate equally likely in both the negative and the positive direction. In short, we strengthened the role of the scenario player by removing the Γ-bound, but weakened it by replacing a malign adversary by a tame distribution. We refer to the Γ_i restricted adversary as the *worst-case* adversary, and to the fully symmetrically distributed, non-Γ_i restricted random variable as the *random* adversary.

For our explanatory purpose maximal generality of the result is less important than insight into the structure of the matter. Therefore we make two simplifying assumptions:

- Let Γ_i be integral.
- Let A be fully symmetrically distributed over $\bigotimes_{i,j}\{a^0_{ij} - \tilde{a}_{ij}, a^0_{ij} + \tilde{a}_{ij}\}$.

As to the first assumption: We will see that a worst-case scenario can be constructed by picking Γ_i entries in each row, on which the scenario deviates maximally either to the negative or the positive direction. In case, Γ_i is not integer, the worst-case adversary can pick $\lfloor \Gamma_i \rfloor$ maximally deviating entries and spend the fractional rest on one further entry. Essentially, the results stay the same. But, keeping track of this fractional rest causes the whole presentation to be a bit clumsy. For this reason we choose to present the weaker claim, i.e., require $\Gamma_i \in \mathbb{N}$.

It can be proven that the second assumption does not affect the generality of the result, but rather represents a worst case for coincidental covering among all fully symmetrical distributions. Yet, to the best of our knowledge, there is no insightful proof for this. Though the proof is technical, the result itself is very intuitive: We seek a bound for the deviation in the matrix product Ax. It is not surprising that the deviation is greatest, if the fully symmetric distribution for A has maximal variance, i.e., for each matrix entry the probability mass is centered at the boundary of the interval.

Granted these two simplifications we can sketch a rather short and clear arranged proof of the bound that rests mostly on the same ideas as the one in the appendix of [22]. We only need one small, technical argument, which we prove in Lemma 4.8.

For convenience we argue by the relative deviation $\delta_{ij} := (a_{ij} - a^0_{ij})/\tilde{a}_{ij}$ instead of a_{ij} in the remainder. Note that the second simplification yields the discrete, uniform distribution on the vertices of an $n \times m$-dimensional hypercube for the random variable $(\delta_{ij})_{(ij)}$.

Theorem 4.7 (Simplified Bertsimas-Sim Bound). *Let x be a feasible solution to Problem (4.28) with $\Gamma_i \in \mathbb{N}$ for each row-index i. Further, let $(\delta_{ij})_{(ij)}$ be distributed uniformly over the vertices of the $n \times m$-dimensional hypercube. Then we have:*

$$\Pr\left[(a^0_{ij} + \delta_{ij}\tilde{a}_{ij})_{(ij)}x > b\right] \leq 1 - \left(1 - \left(\frac{1}{2^n}\sum_{\ell=\frac{\Gamma_i+n}{2}}^{n}\binom{n}{\ell}\right)\right)^m$$

Sketch of proof:

1. **Focus on a Single Row.** The worst-case adversary is restricted by $\sum_j \delta_{ij} \leq \Gamma_i$ for each row i independently. And by independence of distribution the random adversary has probability to choose a certain row vector δ_{i*} independent from the entries in other rows. Thus, it suffices to show for each row i that

$$\Pr\left[\sum_j (a^0_{ij} + \delta_{ij}\tilde{a}_{ij})x_j > b_i\right] \leq \left(\frac{1}{2^n}\sum_{\ell=\frac{\Gamma_i+n}{2}}^{n}\binom{n}{\ell}\right).$$

   As we argue for a single row, we will drop the row index in the remainder. Note that, e.g., the variable b is a scalar now.

2. **Adversary's choice.** For a fixed solution x consider the following scenario. Let $I, |I| = \Gamma$, be an index set with $\tilde{a}_j|x_j| \geq \tilde{a}_\ell|x_\ell|$, for all $j \in I$ and $\ell \notin I$. Let $\delta_j = \text{sign}(x_j)$ for all $j \in I$ and zero else. This is a possible strategy for the worst-case adversary, which maximizes $\delta_j\tilde{a}_jx_j$, because it respects the Γ-bound. Intuitively, the set I contains those entries where the adversary can cause greatest damage.

   Thus, if any scenario, then this will make x violate: $\sum_j \delta_j\tilde{a}_jx_j \leq b - \sum_j a^0_jx_j$. We call it the *test scenario* δ^x for x. The name is justified, because it is a Γ-bounded scenario, and if an x is feasible in its test scenario, then it is feasible in all Γ-bounded scenarios.

3. **The Test Vector.** Substitute $t_j := \tilde{a}_jx_j$ and $\delta^t := \delta^x$. Then calculating the random adversary's probability to make a robust solution fail, boils down to the following question: For an n-dimensional vector t, how many of the 2^n vectors $\delta \in \{-1, 1\}^n$ fulfill

$$t'\delta > t'\delta^t.$$

   By Lemma 4.8 we know that this number is greatest for the vector $t^1 = (1, 1, \ldots, 1)$.

4. **Counting Vertices.** It remains to count the number of vectors $\delta \in \{-1, 1\}^n$ with $\delta't^1 > \delta^{t^1\prime}t^1 = \Gamma$. These are exactly those $\delta \in \{-1, 1\}^n$ which have more than Γ more positive than negative entries. Let p denote the positive and q the negative entries of such a vector δ. Then $p - q = p - n + p > \Gamma$ gives $p > (\Gamma + n)/2$, and we get for the number of such vectors in $\{-1, 1\}^n$:

$$\sum_{\ell=\frac{\Gamma_i+n}{2}}^{n} \binom{n}{\ell}$$

which yields the result.

Apart from the simplifications the simplified Bertsimas-Sim bound seems not to be the same as the theorem found in [22]. And it seems that the simplified theorem we give here, does not show the asymptotic behavior promised earlier. But this is only true at first glance. Bertsimas and Sim do not state an explicit bound on the probability for the whole *system* of inequalities to be fulfilled. They state the bound for a single row. For a single row, the probability bound to be infeasible decreases rapidly. For a single row our theorem is a simplification of that in [22]. For a single row the Γ-bound yields reliable solutions. But for the whole system, as shown in our theorem, the bound behaves badly with increasing number of rows m. Again this must be the case, *because strict robustness treats each row separate.* Even a small, independent probability for each single inequality to be violated, yields a considerable probability for a system of many inequalities to contain one which is violated. The Bertsimas-Sim bound is not a bound on the probability for feasibility of an inequality system but for feasibility in one inequality.

Sparse Matrices and Few Rows The above consideration motivate again to complement the horizontal integration by the Γ-bound with at vertical integration also from the perspective of coincidental covering, as we will do in the remainder. Still, the Bertsimas-Sim model is a successful model for special types of problems. We want to distinguish in more detail problems for which the model is suitable from those for which it is not.

When a coefficient a_{ij} in a linear program equals zero, it is often only an artefact of modeling. There is no relation in the modeled reality that corresponds to that coefficient. It rather means that the j-th variable has no influence to the i-th constraint at all. For example, the i-th constraint may express that the total amount of alcohol measured in gram contained in a diet must not be larger than some threshold. And the j-th variable may model the amount of meat measured in gram assigned by the diet. The exact amount of alcohol contained in one gram of wine is subject to uncertainty, but the amount of alcohol in one gram of meat is by all reasonable means equal to zero. Zero coefficients do not express input data subject to changes, but a fixed structure of the model.

This can still be captured in the approach of [22]. We can fix $\tilde{a}_{ij} = 0$ for those entries. Still, this gives a problem, if the matrix is sparse. Setting the interval length equal to zero for almost all entries in a row, deprecates the Γ-bound. If there are for example only two entries subject to changes, than any $\Gamma \geq 2$ is superfluous. In this case the model will not be less conservative than the Soyster model.

A typical example for sparse matrices are problems with incidence- or adjacency matrices of a graph as part of the linear model. These matrices have entries in $\{0, 1\}$ or $\{-1, 0, 1\}$. Usually not only the zero entries but all entries of the rows modeling the graph are assumed to be invariant for all scenarios. In this case we must distinguish two types of models. For one of these the Bertsimas-Sim approach and the bound are particularly well suited. Among the well suited models are flow models in which the *uncertainty is only in the cost function.* As a typical example, consider a shortest path problem with uncertain arc-length [21]. Uncertainty in a linear cost function c is modeled by an extra variable C and an extra row $c'x \leq C$ (for a minimization problem). In this case the new row is the only one affected by uncertain data. By the above consideration, of course, a single uncertain row with a lot of uncertain coefficients is the perfect situation for the Bertsimas-Sim bound.

Quite the opposite is true, if the uncertainty is completely in the right-hand side. Prima facie, right-hand side uncertainty seems the easier situation. But by the techniques developed in classical robust optimization, one can do no better than replacing the right-hand side of each inequality by the largest (for minimization problems the smallest) value that can occur in any scenario. This will yield very conservative solutions. Note that the robust timetabling problem is exactly of that kind.

Still, the basic idea of Bertsimas and Sim to limit the number or the total amount of disturbances in each scenario is convincing in this context, too: Usually, only a few right-hand side entries will change. But, this point of view to the scenario is necessarily a vertical integration: One has to consider the disturbances in all rows simultaneously. This cannot be accomplished by the classical approach. The at first sight simpler case of right-hand side uncertainty requires the concept of recovery robustness.

We now prove the lemma reducing the test vector to the all-one vector. The first half of the proof is similar to the proof of Proposition 2 in [22]. The claim of the lemma is not surprising: if person A choose the Γ largest entries, and person B chooses some entries at random, then person B will have best chances to get a bigger sum than B, if the entries are all equal.

Lemma 4.8. *Let δ be an n-dimensional random variable, uniformly distributed over $\{-1, 1\}^n$, and t be some n-dimensional vector, with I, $|I| = \Gamma$, an index set of maximal entries in $|t|$, i.e., $|t_j| \geq |t_\ell|$ for all $j \in I$ and $\ell \notin I$.*

Further, define $\delta^t_j = \operatorname{sign}(t_j)$ *for all* $j \in I$ *and zero else. Then*

$$\Pr[t'\delta > t'\delta^t] \leq \Pr[\|\delta\|_1 > \Gamma].$$

Proof. We first scale the vector by the value of the Γ-largest entry, and set all of the Γ largest equal to 1. In the second step, we lift the entries smaller than 1 up, starting from the smallest.

Lifting the smallest entry will make the scalar product of the random vector in some cases smaller than the threshold and in some cases larger. We show that the later is more likely.

Let $I^c := [n] \setminus I$ be the complement of I in $[n]$, and denote by $t_* := \min_{j\in I} t_j$.

We have $\Pr[t'\delta > t'\delta^t] = \Pr[\sum_{j\in[n]} |t_j|\delta_j > \sum_{j\in I} |t_j|]$ by symmetry of distribution.

$$\begin{aligned}
\Pr\left[\sum_{j\in[n]} |t_j|\delta_j > \sum_{j\in I} |t_j|\right] &= \Pr\left[\sum_{j\in I^c} |t_j|\delta_j > \sum_{j\in I} |t_j|(1-\delta_j)\right] \\
&\leq \Pr\left[\sum_{j\in I^c} |t_j|\delta_j > |t_*|\sum_{j\in I}(1-\delta_j)\right] \\
&= \Pr\left[\sum_{j\in I^c} |t_j|\delta_j + |t_*|\sum_{j\in I}\delta_j > |t_*|\sum_{j\in I} 1\right] \\
&= \Pr\left[\sum_{j\in I^c} \frac{|t_j|}{|t_*|}\delta_j + \sum_{j\in I}\delta_j > \Gamma\right]
\end{aligned}$$

Let T_Γ be the set of all vectors $t \in \mathbb{R}^n_{\geq 0}$ where the Γ maximal entries are all equal to 1. We have shown that the maximal value for $\Pr[t'\delta > t'\delta^t]$ over all t $\in \mathbb{R}^n$ is attained by some $t \in T_\Gamma$. Proposition 2 in [22] gives a similar statement. Now, we show that among all $t \in T_\Gamma$ the all-one vector maximizes $\Pr[t'\delta > t'\delta^t]$. Note that for all $t \in T_\Gamma$ we have $t'\delta^t = \Gamma$.

Let $t \in T_\Gamma$. We will successively lift the smallest entry t_i to 1 until we end up with the all-one vector. Formally, for any $t \in T_\Gamma$ with smallest entry t_i we set $\bar{t}_j = t_j$ for all $j \neq i$ and $\bar{t}_i = 1$. We partition the scenario space for all entries except the smallest, the i-th entry:

$$P_1 := \Pr\left[\sum_{j\neq i} t_j\delta_j \in (-\infty, \Gamma - t_i]\right],\ P_2 := \Pr\left[\sum_{j\neq i} t_j\delta_j \in (\Gamma - t_i, \Gamma - t_i]\right],$$

$$P_3 := \Pr\left[\sum_{j\neq i} t_j\delta_j \in (\Gamma + t_i, \Gamma + 1]\right],\ \text{and } P_4 := \Pr\left[\sum_{j\neq i} t_j\delta_j \in (\Gamma + 1, \infty)\right].$$

What is the probability that $\bar{t}'\delta > \Gamma$ but $t'\delta \leq \Gamma$? The answer is P_3 times the probability for $\delta_i < 0$, i.e., $\delta_i = -1$. Likewise, the probability that $t'\delta > \Gamma$ but $\bar{t}'\delta \leq \Gamma$ equals $P_2 \cdot \Pr[\delta_i = 1]$. As δ_i is distributed symmetrically it remains to show that $P_3 \leq P_2$.

Take a vector δ such that $\sum_{j \neq i} t_j \delta_j \in (\Gamma + t_i, \Gamma + 1]$ holds. For at least one $\ell \notin I$ we have $\delta_\ell = 1$. Define δ^ℓ to equal δ except for $\delta^\ell_\ell = -\delta_\ell$. Then, as $t_i \leq t_\ell \leq 1$ we get for $\delta^\ell : \sum_{j \neq i} t_j \delta^\ell_j \in (\Gamma - t_i, \Gamma - t_i]$. Therfore, we have $P_2 \geq P_3$, which yields the lemma.

□

Summary The core of the Bertsimas-Sim bound is a geometric insight. On the one hand, we look at the 1-ball in the infinity-norm, i.e., the n-dimensional hypercube $B_{(\infty,1)} = \{z \in \mathbb{R}^n : \|z\|_\infty \leq 1\}$ and on the other hand, we consider the intersection of $B_{(\infty,1)}$ with the Γ-ball in the 1-norm, $B_{(1,\Gamma)} = \{z \in \mathbb{R}^n : \|z\|_1 \leq \Gamma\}$. For a fixed vector $t \in \mathbb{R}^n$ that has a scalar product less than some constant C with each vector in $B_{\infty,1} \cap B_{(1,\Gamma)}$, we analyzed the maximal number of vertices of the cube, respectively the mass of vectors in $B_{\infty,1}$, which also have scalar product with t less than C. Assuming a fully symmetric distribution on the hypercube (respectively its vertices) allows to express this analysis in a compact way, namely as an upper bound on the probability of a particular event.

Of course, this particular event is an important event, namely the event that a robust solution is infeasible. And, of course, the distribution is more than a tool to express some analysis in a compact way. The distribution turns the analysis into an interesting and useful result, because fully symmetric distributions are a reasonable assumption in some applications. Yet, the approach of [22] has been criticized exactly because of these assumptions on the distribution. In many applications assuming symmetrically distributed data is not justified. Nevertheless, the criticism seems unfair. By a closer look at the proof for the bound, in particular, at the geometric core of this argument, it becomes conceivable that similar results can also be proven for non-symmetric distributions. So, it rather seems a dilemma of presentation: Assuming properties of the distribution yields relevant results, but makes the analysis seemingly less general than it actually is.

One can avoid this dilemma by working with the assumption that one has no knowledge about the distribution. One special way to do so, is to assume a black-box distribution, i.e., that the distribution is only given by a sample of scenarios, drawn with respect to that distribution. We will discuss this setting briefly in the sequel.

We take a different approach: In a concrete application one may have substantial knowledge about the distribution, and this knowledge shall be exploited. So, the general tool provided in advance must hold for different types of distributions, and must contain an interface to plug in the concrete type of distribution. Together, the general result and the specific knowledge about the distribution give the full result. Therefore, in Section 4.5 we will concentrate on the geometric insight to coincidental covering by recovery robust solutions. Given a concrete probability measure, those results can be used as lemmata for lower bounds on the probabilistic mass that is coincidentally covered.

Results for the black-box model are useful, if the distribution is not known in the application. Our results are useful, if the distribution is known in the application, although it is not known to the author of this text.

4.3.5 Black-box Distributions

One way to avoid the presentation dilemma is to assume a so-called *black-box distribution* (cf. [53] for a recent, exemplary result). In a black-box distribution model, one assumes to have no knowledge of the distribution, except for the possibility to draw some samples. Figuratively, the distribution is inside a black-box. We can only learn about it by observing it for some time. For a 2-stage stochastic linear program the so-called *Sample Average Approximation* (SAA) method [48, 38] is in a certain sense the best one can do. In particular, the part of [53] solving a 2-stage stochastic linear program can be replaced by the simpler SAA method [49].

The SAA method is strikingly simple:

- **Optimal Solution:** Given sufficient computational power, full knowledge of the distribution, and assuming a discrete scenario set, the optimal solution of a 2-stage stochastic linear program can be found by solving a huge linear program, namely the scenario expansion. The constraint matrix of this program has a block structure, one block for each scenario. Each block models the situation in one scenario. Thus, one part of the variables is fixed for all scenario blocks, i.e., the first stage decision. Each second stage variable belongs to a scenario, only occurs in that scenario's block, and is weighted in the objective function by that scenarios probability.

- **SAA-Solution:** We neither have full knowledge of the distribution, nor the computational power needed for solving the scenario expansion. We can only sample a number of scenarios. As this is all we know about the distribution, let us treat these samples, as if they were the whole distribution: Construct the scenario expansion for the sampled scenarios, and weigh the second stage cost of a scenario by the frequency with which it occurs in the sampling. In short, the SAA method substitutes the whole, unknown scenario space by the small, and known sample set.

- **Quality of SAA:** For rather mild assumptions on the 2-stage stochastic program one can show [38] the following: For a fixed approximation factor α the probability that the result of the SAA method is not an α-approximation for the full scenario expansion, decreases exponentially with increasing number of samples. The most general conditions given for this type of result are quite technical. An important corollary states that the claim holds, if the objective value is a Lipschitz-continuous function of the solution. In [48] it is also noted that the approximation factor will in general *not* decrease rapidly with increasing sample size.

- **Idea of Proof:** The actual proof is rather technical. Still, the idea is straight forward. For a first stage solution x consider the scenario set partitioned in the following way. Construct the partition such that scenarios in the same set cause almost equal cost for x—by a precision depending on α and the Lipschitz-coefficient. The cost caused for x in the first stage is equal for all scenarios. Thus, the partition groups those scenarios together, for which the second stage cost of an optimal second stage decision for x is alike. This partition will not contain very small sets due to the Lipschitz-condition. Hence, the sets of the partition will be adequately represented in a sufficiently large sample with high probability. Therefore, the SAA method will price the solution x almost in the same way as the full scenario expansion will price it.

- **Artefacts:** In the spirit of the above argument, let α and the Lipschitz-coefficient cause a partition of the scenario set into M sets. What happens if the sample size is smaller than M? Then the SAA method will produce solutions that still fit perfectly well to the sampled scenarios, but might be completely inappropriate for those parts of the scenario space not represented in the sample set. The solution then becomes an artefact of the sample set, and not necessarily a good solution to the real problem.

The SAA method is tailored for black-box distributions. But, if the sample-size is too small, it cannot provide for a good solution. The information contained in a small sample set is simply not sufficient for finding a good solution. One may argue that this is not the fault of the SAA method, which makes best use of the available data. Still, the question is, whether in practice one really faces a black-box situation.

For delay-resistant timetabling, one can describe fairly precise distributions for the disturbances on each arc. These are the marginal distributions of the distribution for the whole railway system. There are dependent disturbances, e.g., those due to bad weather conditions. Other types of disturbances are modeled well by an independent random variable for each arc. While the first type of scenarios can be modeled by a small scenario set representing the few different weather conditions, thus yielding a model that can be solved by the scenario expansion, the independent type of disturbance give extremely large scenario sets even for problems with no more than 20 arcs.

In this case we face a huge scenario space for which we know the distribution completely, but implicitly by independent marginal distributions. We argue that in this situation it is not wise to ignore the structural knowledge about the distribution and to restrict to a sampling approach. The sample size will easily be too small and the available information wasted. We rather seek a model that is able to exploit the given, rich, structural knowledge.

4.3.6 The Scenario Sets

Motivated by the discussion in the tradition of robust optimization we choose the following types of scenario sets. We define them with respect to a reference scenario (A^*, b^*). Usually this is equal to the nominal system of inequalities (A^0, b^0). We first give the scenarios set for right-hand side uncertainty:

$$S^{\mathbf{\Delta}} := \{b \in \mathbb{R}^m : |b_i - b_i^*| \leq \mathbf{\Delta}_i\}$$

$$S^{\Delta_1} := \left\{ b \in \mathbb{R}^m : \sum_i |b_i - b_i^*| \leq \Delta_1 \right\}$$

By scaling the rows we can assume w.l.o.g. that all disturbances have the same bound $\mathbf{\Delta_i} = \Delta_\infty$, unless some entry of $\mathbf{\Delta}$ is equal to zero. So we define:

$$S^{\Delta_\infty} := \{b \in \mathbb{R}^m : |b_i - b_i^*| \leq \Delta_\infty\}.$$

Let $k : \frac{\Delta_1}{\Delta_\infty}$ and define

$$S^k := S^{\Delta_1} \cap S^{\Delta_\infty}.$$

In delay resistant timetabling we face right-hand side disturbances. Moreover, we should assume them to be *hostile* in the sense that the data deviates from the reference scenario only in direction which tightens the restriction, i.e., in negative direction for an inequality of the form $a'x \leq b$ and in positive direction for $a'x \geq b$. This hostility only plays a role, when it comes to prove lower bounds for the mass of the coincidentally covered scenarios. Recall from the above analysis that the non-hostility is essential for the Bertsimas-Sim bound. We will prove lower bounds for the hostile case, which are automatically also lower bounds to the non-hostile case. Still, the non-hostile case is not convincing, at least for right-hand side disturbances. It is not a convincing argument for an optimization method to stress that the mathematically found solution stays feasible, even when things turn out better than expected.

For disturbances in the matrix we follow the idea of vertical integration. So our scenario sets are of the form:

$$S^{\bar{k},k} := \left\{ A = A^* - \tilde{A} \in \mathbb{R}^{m,n} : \tilde{A} \in \tilde{S}^{\bar{k},k} \right\}$$

with

$$\tilde{S} := \tilde{S}^{\bar{k},k} := \left\{ (\tilde{a}_{ij})_{(ij)} \in \mathbb{R}^{m,n} : \sum_j \tilde{a}_{i,j} \leq \bar{k}, \sum_{i,j} \tilde{a}_{i,j} \leq k, 0 \leq \tilde{a}_{i,j} \leq \Delta_\infty \right\}$$

By the same rationale as above we can assume hostile disturbances also for the case of uncertain data in the planning matrix.

In some parts we will set $\Delta_\infty = 1$ for the sake of simpler presentation.

In order to avoid confusion we will restate the definition and notation of the scenario sets whenever they are used.

4.4 Linear Recovery Robust Programs

This section and the next section combine the broad notion of recoverable robustness and the inspiration from the tradition of robust programming. In this section we mainly show how LRPs can be solved for certain scenario sets. This leads us to a special case, namely robust network buffering, which entails the robust timetabling problem. Towards the end of this section we turn to a variant of LRPs, where the planning problem is an *integer* linear program.

4.4.1 Solving Right-hand Side LRPs

In this part we show that some scenario sets for the right-hand side data of an LRP yield problems that can be solved by a relatively small linear program.

Consider again the 3-player formulation of an LRP (4.21). Let $P := \{x \in \mathbb{R}^n : A^0 x \geq b^0\}$ be the polytope of nominally feasible solutions. If we fix the strategies of the first two players, i.e., the variables x and (A, b), we get the recovery problem of the LRP: $\min d'y$ subject to $\hat{A}y \geq b - Ax$. The dual of the latter is $\max_{\zeta \geq 0}(b - Ax)'\zeta$ s.t. $\hat{A}'\zeta \leq d$. The recovery problem is a linear program. Thus, we have strong duality, and replacing this linear program by its dual in expression (4.21) will not change the problem for the players optimizing x respectively (A, b).

$$\begin{aligned}\min_{x \in P} c'x \text{ s.t. } D &\geq \left\{\max_{(A,b)\in S}\left\{\max_{\zeta \geq 0}(b - Ax)'\zeta \text{ s.t. } \hat{A}'\zeta \leq d\right\}\right\} \Leftrightarrow \\ \min_{x \in P} c'x \text{ s.t. } D &\geq \left\{\max_{(A,b)\in S, \zeta \geq 0}(b - Ax)'\zeta \text{ s.t. } \hat{A}'\zeta \leq d\right\} \qquad (4.33)\end{aligned}$$

Consider the maximization problem in formulation (4.33) for a fixed x, thus find $\max_{(A,b)\in S, \zeta \geq 0}(b - Ax)'\zeta$ subject to $\hat{A}'\zeta \leq d$. Assume for a moment $\|b - Ax\|_1 \leq \Delta$. In this case, for each fixed vector ζ the maximum will be attained, if we can set $\operatorname{sign}(\zeta_i)(b - Ax)_i = \Delta$ for i with $|\zeta_i| = \|\zeta\|_\infty$ and 0 otherwise. In other words, under the previous assumptions $(b - Ax)'\zeta$ attains its maximum when $(b - Ax) = \Delta e_i$ for some suitable $i \in [m]$. Therefore, if we have $\|b - Ax\|_1 \leq \Delta$, we can reformulate problem (4.33):

$$\begin{aligned}\min_{x \in P} c'x \text{ s.t. } \forall i \in [m] : D &\geq \left\{\max_{\zeta \geq 0}(\Delta e_i)'\zeta \text{ s.t. } \hat{A}'\zeta \leq d\right\} \Leftrightarrow \\ \min_{x \in P} c'x \text{ s.t. } \forall i \in [m] : D &\geq \left\{\min_{y} d'y \text{ s.t. } \hat{A}y \geq \Delta e_i\right\} \qquad (4.34)\end{aligned}$$

The at first sight awkward condition $\|b - Ax\|_1 \leq \Delta$ is naturally met if only the right-hand side data changes, and is limited in the set $S^1 = \{(A^s, b^s) : \|b^* - b^s\|_1 \leq \Delta, A^* = A\}\}$. For an LRP over S^1 formulation (4.34) is equivalent to a linear program of size $\mathcal{O}(m(n + \hat{n} \cdot m))$:

$$\begin{aligned} \min_{x \in P} \quad & c'x \\ \text{s.t.} \quad & \forall i \in [m]: \\ & \qquad Ax + \hat{A}y^i \geq b + \Delta e_i \\ & \qquad d'y^i - D \geq 0 \end{aligned}$$

Next, consider the scenario set S^k for arbitrary $k > 1$. By the same token, the maximization over ζ in formulation (4.33) for fixed x and (A, b) can be achieved, by setting the maximal $\lfloor k \rfloor$ entries of the vector $(b - Ax)$ equal to 1 and the $\lceil k \rceil$-th entry equal to $k - \lfloor k \rfloor$. For example, when k is integer, we can replace the scenario set S^k by those $\binom{k}{m}$ scenarios, where exactly k entries of b deviate maximally from b^*, and the other entries equal their reference value b^*_i. So, we have:

Theorem 4.9. *An LRP over S^k can be solved by a linear program of size polynomial in n, $\hat{n}$, m, and $\binom{k}{m}$.*

Corollary 4.10. *An LRP over S^1 can be solved by a linear program of size polynomial in n, $\hat{n}$, and m.*

Corollary 4.11. *For fixed k an LRP over S^k can be solved by a linear program of size polynomial in n, $\hat{n}$, and m.*

Of course, in practice this approach will only work, when k is very small.

The above reasoning can give a fruitful hint to approach RROPs in general. First, try to find a small subset of the scenario set, which contains the worst-case scenarios, and then optimize over this set instead of the whole scenario set. In the above setting we can achieve this very easily, because the recovery problem fulfills strong duality. If the recovery problem is an integer program this approach fails in general. Still, one can try to find a small set of potential worst case scenarios, to replace the original scenario set. Unlike the recovery problem, the planning problem may well be an integer or mixed integer program, as we show in the following.

For the manipulations of the formulations the linearity of $P, Ax \geq b$ or c is immaterial. So we can extend the above reasoning to non-linear optimization problems. Let $c : \mathbb{R}^n \to \mathbb{R}$ be a real function, P' a set of feasible solutions and $\{g_i : \mathbb{R}^n \to \mathbb{R}\}_{i\in[m]}$ be a family of real functions, and assume that extrema in the resulting RROP are either attained, or the problem is unbounded. For the scenario set S^1 of right-hand side disturbances we have with the above notation

$$\begin{gathered}
\min_{x\in P'} c(x) \\
\text{s.t. } D \geq \left\{\max_{b\in S^1}\left\{\min_y d'y \text{ s.t. } g(x) + \hat{A}y \geq b\right\}\right\} \\
\Leftrightarrow \\
\min_{x\in P'} c(x) \\
\text{s.t. } \forall i \in [m] : g(x) + \hat{A}y^i \geq b^* + \Delta e_i \\
D - d'y^i \geq 0
\end{gathered}$$

because the recovery problem is still a linear program, and we can thus argue by strong duality as in the previous case. In particular, we are interested in the case of an integer linear program $(\min c'x, Ax \geq b, x \in \mathbb{Z}^n)$ with right-hand side uncertainty. We get as its recovery robust version over S^1.

$$\begin{gathered}
\min_{x\in\mathbb{Z}^n} c'x \\
\text{s.t. } A^0x \geq b^0 \\
D \geq \left\{\max_{b\in S^1}\left\{\min_y d'y \text{ s.t. } A^*x + \hat{A}y \geq b\right\}\right\} \\
\Leftrightarrow \\
\min_{x\in\mathbb{Z}} c'x \\
\text{s.t. } A^0x \geq b^0 \\
\forall i \in [m] : A^*x + \hat{A}y^i \geq b^* + \Delta e_i \\
D - d'y^i \geq 0
\end{gathered}$$

Let $A^* = A^0$ and $b^* = b^0$. Defining $f := A^*x - b^*$ we can rewrite the previous program as

$$\begin{gathered}
\min_{(x,f)\in\mathbb{Z}^{n+m}} \tilde{c}'(x,f) \\
A^*x - f = b^* \\
\text{s.t. } \forall i \in [m] : f + \hat{A}y^i \geq \Delta e_i \\
d'y^i - D \geq 0 \\
f \geq 0
\end{gathered} \tag{4.35}$$

With a suitable cost vector $\tilde{c}$. Note that the original integer linear program corresponds with the scenario part of the program only via the slack variable f. In other words, for solving the recovery robust version the solving procedures for the original, deterministic, integer linear optimization problem can be left untouched. We only have to flange a set of linear inequalities to it. The f variables function as means of communication between the original integer problem, where they correspond to the slack in each row, and the linear part, in which their effect on robustness is evaluated. In the next part we will consider this communication situation for an even more specialized type of recovery.

4.4.2 Robust Network Buffering

Let us use Corollary 4.10 for the Simple Robust Timetabling problem with right-hand side uncertainty limited in S^1. Set $g_2 = 0$ to drop the limit to the maximal delay at a node. By the corollary the Simple Robust Timetabling problem over S^1 reads as follows. Let χ_a be the indicator function of a, i.e., $\chi_a(x) = 1$ if $a = x$, and zero else.

$$
\begin{array}{lll}
\min\limits_{\pi,f} & \sum_{e=(i,j)\in E} w(e)(\pi_j - \pi_i) & \\
\text{s.t.} & \pi_j - \pi_i + f_e = t(e), & \forall e = (i,j) \in E \\
\forall s \in E : & & \\
& f_e + y^s_j - y^s_i \geq \Delta \cdot \chi_e(s), & \forall e = (i,j) \in E \\
& D - d'y^s \geq 0 & \\
& f, y^s \geq 0 &
\end{array}
\tag{4.36}
$$

Periodic Timetabling Many railway service providers operate periodic schedules. This means that equivalent events, e.g., the departure of the trains of a line at a station take place in a regular manner. More precisely, we do not plan the single events as in the aperiodic case, but we plan periodic events. For these we schedule a periodic time, which is understood modulo the period of the system. (There may also be differnt periods in the same system, but we restrict our consideration here to the case of a single, global period.) Assume we assign 5 to the variable corresponding to the periodic event that trains of line A depart from station S towards station S'. Let the period T of the system be one hour. Then for every hour five minutes past the hour a train of line A will depart from station S towards station S'.

This leads to the Periodic Event Scheduling Problem (PESP)[1], which can be formulated as a mixed integer program of the following form. Let $G(A, V)$ be a directed graph and three functions $w, u, l : A \to \mathbb{R}$ on the arc set. Then the following problem is called a PESP.

$$\begin{array}{ll} \min\limits_{k \in \mathbb{Z}^{|A|}, \pi} & \sum_{e=(i,j) \in A} w(e)(\pi_j - \pi_i + k_e T) \\ \text{s.t.} & u(e) \geq \pi_j - \pi_i + k_e T \geq l(e), \quad \forall e = (i,j) \in A \end{array}$$

This type of problem has a broad modeling power. For a comprehensive study on periodic timetabling we refer the interested reader to [42].

To construct an RROP from an original problem which is a PESP we have to make a choice whether we interpret the disturbances as periodic disturbances like a construction site that will slow down the traffic at a certain point for the whole day, or as aperiodic events like a jammed door at a stopping event. For periodic disturbances we get the following program.

$$\begin{array}{lrl} & \min\limits_{k \in \mathbb{Z}^{|A|}, \pi, f} & \sum_{e=(i,j) \in A} w(e)(\pi_j - \pi_i + k_e T) \\ & \text{s.t.} & \pi_j - \pi_i + f_e + k_e T = l(e), \qquad \forall e = (i,j) \in A \\ & & \pi_j - \pi_i + \bar{f}_e + k_e T = u(e), \qquad \forall e = (i,j) \in A \\ \forall s \in A, \Xi \in \{0,1\} : & & \\ & & f_e + y_j^s - y_i^s \geq \Delta \cdot \chi_e(i) \cdot \Xi, \qquad \forall e = (i,j) \in A \\ & & \bar{f}_e + y_i^s - y_j^s \geq \Delta \cdot \chi_e(i) \cdot (1 - \Xi), \quad \forall e = (i,j) \in A \\ & & D - d' y^s \geq 0 \\ & & y^s \geq 0 \end{array}$$

Note that the right-hand sides are still constants, though they look like a quadratic term.

Again, the deterministic PESP instance can be flanged with a polynomial size linear program to ensure robustness. This structure can be helpful for solving such a problem, as the specialized solving techniques for the original integer program can be integrated.

As an example for this approach confer [25], where a specialized technique for an advanced platforming problem was combined with robust network buffering to get a recovery robust platforming.

[1]The Periodic Event Scheduling Problem was introduced in [52]. For details confer also [42].

General Network Buffering The general situation is the following: We are given an optimization problem on a network. The solution to that problem will be operated under disturbances. The disturbances propagate through the network in a way depending on the solution of the optimization problem. The solution of the original optimization problem x translates into a buffer vector f on the arcs of the network. Changing perspective, the original problem with its variables x is a cost oracle: If we fix a certain buffering f, the optimization will construct the cheapest x vector to ensure the buffering f. Let us summarize the general scheme.

Given an optimization problem P with the following features:

- A directed graph G.
- An unknown, limited, nonnegative vector of disturbances on the arcs, or on the nodes, or both.
- The disturbances cause costs on the arcs, or on the nodes, or both, which propagate through the network.
- A vector of absorbing potential on the arcs, the nodes, or both can be attributed to each solution of P.

If we further restrict the disturbance vector to lie in S^1, we get the following by the above considerations: The recovery robust version of P, in which the propagated cost must be kept below a fixed budget D, can be formulated as the original problem P plus a linear program quadratic in the size of G.

The Scenario Set S^1 With S^1 as scenario set the fractional network buffering problem can be solved efficiently by a particularly simple LP-based approach. But, the scenario set S^1 is not is not a convincing choice with respect to many applications. By S^1 we only impose a limit to the total amount of disturbance. And the analysis shows that for every solution the worst case is a scenario in which the complete disturbance is concentrated at a single arc. From a practical point of view, one would rather like to use a set where the individual disturbance is smaller, but several of them can occur.

Remember that S^k is defined by two bounds, Δ_1 for the 1-norm, and Δ_∞ for the infinity-norm. Their ratio is k. For fixed Δ_1, what is the difference between the worst case scenarios with $\Delta_\infty = 1$ and those with $\Delta_\infty = 2$? Assume the network to be very large. Say there is an arc where a single disturbance of size 2 can cause a total delay of D_0. Now, place two subsequent disturbances of size 1, one at that arc and the other on one of its immediate successors or predecessors. This will cause a total delay that is only slightly smaller than D_0, if the area affected by the second chosen arc is almost the same, as that affected by the first arc. So, in some network topologies the effect of two small disturbances is almost the same as that of one with twice the size.

Therefore, it makes sense to use the scenario set S^1 and the resulting, simple models with a larger Δ_1. *Instead of protecting against k small disturbances of size* 1*, we protect against one disturbance of size k.*

Note that as a by-product we get a better probability that our interval bound Δ_∞ is satisfied, because $\Delta_\infty = \Delta_1$ in S^k.

A larger Δ_1 is prima facie more conservative. Indeed, for strict robustness every increase of Δ_1 will affect heavily the objective value. But, when the above reasoning about several small disturbances causing almost the same delay as one large disturbance holds true, then recoverable robust will not produce unacceptably conservative solutions, if Δ_1 is increased.

The question is, for which scenarios outside S^1 a solution, which is constructed for S^1, is still recoverable. This is the central question of the last section of this chapter. We will show that one can give an a priori answer to this question. This provides an orientation to trim the choice of Δ_1 and k when constructing the model, in order to keep a balance between the size of the linear program to solve, and the nominal objective of the robust solution.

4.4.3 Integer Robust Network Buffering

The above scheme motivates to consider problems of the following type.

Definition 4.12 (RABP)**.** *Given a directed graph G, a budget D and a vector $c \in \mathbb{R}^{|E(G)|}$. The linear program*

$$\begin{aligned} &\min\nolimits_f c'f \\ s.t.\ \forall s \in E : &f_s + y^s_j - y^s_i \geq \Delta \cdot \chi_e(s), \quad \forall e = (i,j) \in E \\ &D - d'y^s \geq 0 \\ &f, y^s \geq 0 \end{aligned}$$

is called the Robust Arc Buffering Problem *(RABP).*

In a similar way we can define a problem of buffering the vertices of the graph:

Definition 4.13 (RVBP)**.** *Given a directed graph G, a budget D and a vector*

$c \in \mathbb{R}^{|V(G)|}$. *The linear program*

$$\begin{aligned} &\min\nolimits_f c'f \\ s.t.\ \forall s \in E : &f_j + y_j^s - y_i^s \geq \Delta \cdot \chi_e(s), \quad \forall e = (i,j) \in E \\ &D - d'y^s \geq 0 \\ &f, y^s \geq 0 \end{aligned}$$

is called the Robust Vertex Buffering Problem *(RVBP).*

The dual of the RABP has a structure close to that of a min-cost flow problem. We next look at the integer versions of RVBP and RABP, i.e., the above problems with the additional restriction that f must be an integer. It turns out that these problems even for $\Delta^1 = 1$ and $c = (1, 1, \dots 1)$ are NP-hard, as difficult to approximate as vertex cover, and have an arbitrarily large integrality gap.

We also want to introduce the budgeted version of RVBP and RABP. By this we mean that we exchange the role of D and $c'f$. Instead of fixing the maximal delay for all scenarios, we minimize the maximal delay for a fixed buffer budget, i.e., we require $B \geq c'f$.

Theorem 4.14. *The Budgeted Integer RVBP with unit buffer cost and unit disturbance on an acyclic graph is NP-complete in the strong sense and cannot be approximated with a better approximation ratio than the Minimal Vertex Cover Problem.*

Proof. We show the following: Given an oracle providing a buffering with at most α times the cost of an optimal buffering for any instance of the buffering problem, then we can approximate an optimal solution for any instance of Minimal Vertex Cover by a factor of α in polynomial time by a single call of the oracle.

Let $G(V, E)$ be an instance of vertex cover. We construct a buffering instance by the following steps.

1. Choose an arbitrary bijection $o : V \to [|V|]$. Orient every edge $e = \{v, w\} \in E$ to get an arc $a = (v, w)$ such that $o(v) < o(w)$. Obviously, this gives a directed acyclic graph $G_1(V, A)$.

2. Duplicate arcs in $A(G_1)$ in order to achieve equal out-degree for all vertices. Denote the common out-degree of the resulting directed acyclic graph G_2 by d. In case $d = 1$, double all edges. Note that the vertex cover problem on the underlying undirected graph of G_2 is the same as for G.

3. Add a dummy vertex v with $o(v) = 0$ and connect it to all vertices with in-degree 0 applying the same rule for the orientation. The resulting directed acyclic graph G_3 still has the same out-degree.

4. Look for a minimal integer vertex buffering such that the delay is less or equal to $d + 1$.

Any vertex cover for G defines a vertex buffering for G_3 in the obvious way. This is a feasible buffering, i.e., it allows for delay of at most $k + 1$: All arcs—those corresponding to an edge in $E(G)$ and their copies or those originating at the dummy vertex—lead to one of the original vertices, i.e., those in $V(G)$. But these are either in the cover, hence, buffered, or all of their d outgoing edges end in buffered vertices. Therefore, the total delay caused by a single disturbance on such an arc is at most $d + 1$.

For the opposite direction, assume a vertex buffering, where two vertices a, b (with $o(a) < o(b)$) connected in G are not buffered. Then a disturbance on any incoming arc to a will cause at least a delay of $2d$. Thus any feasible vertex buffering of G_3 defines a vertex cover on G.

We have shown that all subsets of $V(G)$ are either a vertex cover for G and a vertex buffering for G_3 or none of both. As no minimal vertex buffering for G_3 buffers the dummy vertex, we know that minimal bufferings and minimal covers have the same vertex sets. This yields the claim.

The construction uses only a polynomial number of parallel arcs. So the RVBP problem is NP-hard in the strong sense.

Obviously, the integer RVBP is in NP. □

We described the reduction for *vertex* buffering. For *arc* buffering we can easily transform the construction. Replace every vertex v in G_3 by a path of length 1, in which the arc (a, b) is called the vertex-arc of v. From this construct a graph G_4 by connecting all in-coming arcs of v with a and all out-going arc of v with b. A buffer on a vertex-arc in G_4 corresponds to the vertex being part of the cover in G. Now, the rationale for the reduction via G_3 carries over to G_4, once we can show that it is possible in polynomial time to transform any minimal buffering of G_4 into a buffering only using vertex-arcs.

To this end, we have to slightly modify step 2 (yielding G_2). Construct G_2' such that every edge in G has at least two corresponding parallel arcs in G_2'. Assume a feasible buffering, which buffers a non-vertex-arc e in G_4'. There is at least one arc e' parallel to e. In case e' is also buffered, push one buffer to the vertex-arc preceding e and e', and the other buffer to the vertex-arc to which they lead. In case e' is not buffered, push the buffer from e to the preceeding vertex-arc. If a vertex-arc, to which we push a buffer, is already buffered, we can remove the pushed buffer completely. It is easy

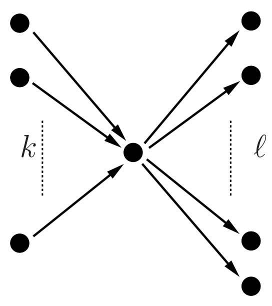


Figure 4.7: Networks with $k + \ell$ arcs producing unbounded integrality gaps.

to check that the resulting buffering is still feasible, at least as cheap as the original, and that it buffers only vertex-arcs. This way we get as a corollary:

Corollary 4.15. *The Budgeted Integer RABP with unit buffer cost and unit disturbance on an acyclic graph is NP-complete in the strong sense and cannot be approximated with a better approximation ratio than the Minimal Vertex Cover Problem.*

Next, we analyze the integrality gap of robust network buffering problems.

Integrality Gap We show that the Integer RABP and the Budgeted Integer RABP, i.e., the problem of finding an arc buffering that minimizes B for fixed D or vice versa, have an unbounded integrality gap. We will denote the integer bufferings by $g : E(G) \to \mathbb{N}$ and the fractional bufferings by $f : E(G) \to \mathbb{R}_{\geq 0}$.

Budgeted Integer RABP: Consider the graph depicted in Figure 4.4.3 with k incoming arcs and ℓ outgoing arcs to the central vertex. Let $k < l$. Choose a value $0 < \alpha < 1$ and fix $D = \lceil \alpha\ell \rceil + 1$.

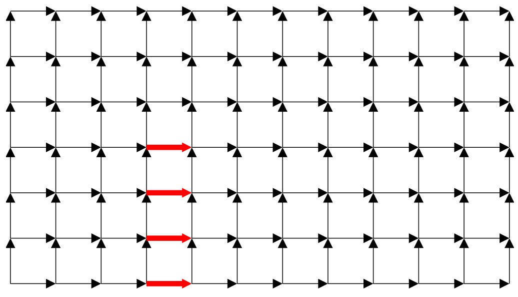
Setting $f(e) = 1 - \alpha$ for each e of the k incoming arcs and zero else, is a feasible fractional buffering with cost $\sum f = k(1 - \alpha)$.

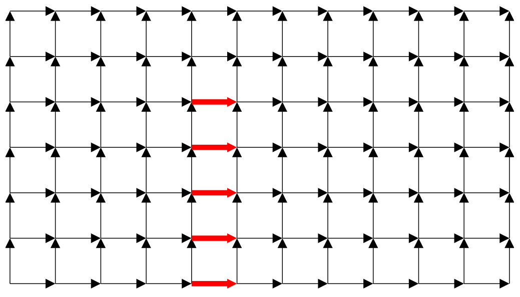
For $k < \lceil (1 - \alpha)\ell \rceil$ an optimal buffering $g(e)$ is equal to 1 on the k incoming arcs e and zero else. For $k \geq \lceil (1 - \alpha)\ell \rceil$ an optimal solution will buffer $\lceil (1 - \alpha)\ell \rceil$ of the outgoing arcs and no incoming arc. The costs of the optimal integer bufferings are in both cases equal to the number of buffered arcs. So for the ratio we either have $\sum g / \sum f \geq l/k$ or equal to $\sum g / \sum f = 1/(1 - \alpha)$.

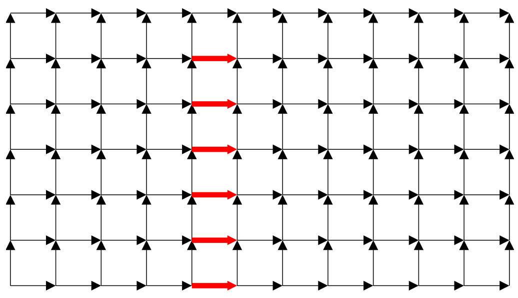
Integer RABP: Consider the same type of graph with $B = k - 1$ fixed, and again $k < \ell$. Then a fractional buffering can spend $(k - 1)/k$ of the buffer budget for each of the k in-coming arcs. Therefore, the maximal total delay of an optimal fractional buffering f is $D(f) = (l + 1)/k$. The integer buffering cannot buffer all in-coming arcs. Therefore, it is best to spend the whole budget on the out-going arcs, yielding a total delay for the integer buffering g of $D(g) = l - k + 2$. The resulting ratio is:

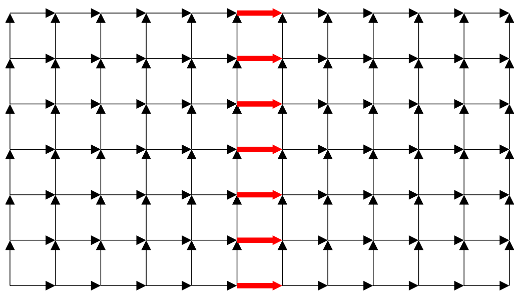
$$\frac{D(g)}{D(f)} = k\left(1 - \frac{k-1}{l+1}\right).$$

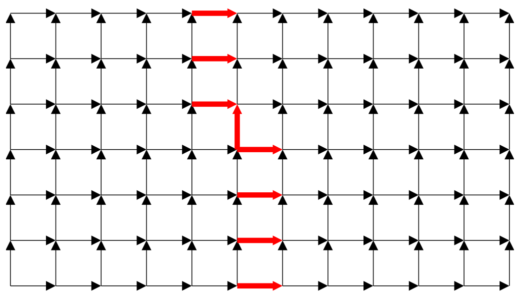
In Figure 4.8 we show a sequence of optimal, budgeted, integral arc bufferings for a unit disturbance and unit weights with increasing budget B for the buffers.

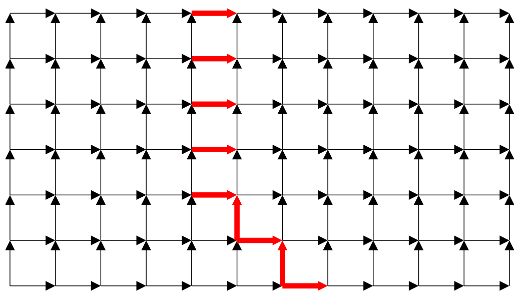

(a) Buffer Budget $B = 4$

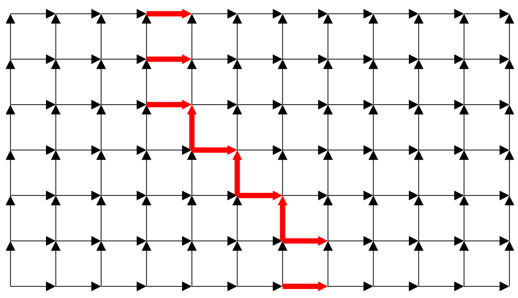

(b) Buffer Budget $B = 5$

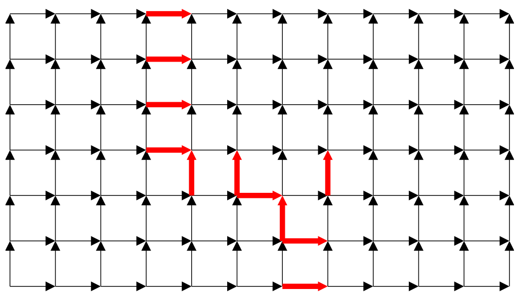

(c) Buffer Budget $B = 6$


(d) Buffer Budget $B = 7$


(e) Buffer Budget $B = 8$


(f) Buffer Budget $B = 9$


(g) Buffer Budget $B = 10$


(h) Buffer Budget $B = 11$

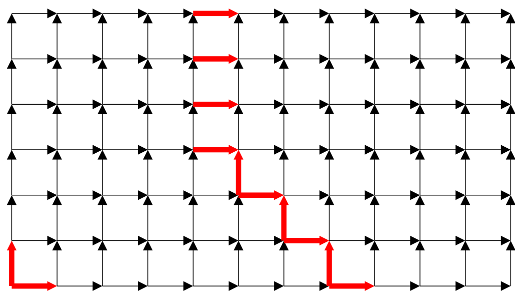

(i) Buffer Budget $B = 12$

Figure 4.8: Optimal, budgeted, integral arc bufferings of a grid.

4.5 A Polyhedral Perspective

The most important result of this section is the perspective in which we look at LRPs. We consider the polyhedral situation in the row space. More precisely, we look at the relations between the image of the admissible recovery vectors on the one side, and the polytope containing all possible vectors that may occur as slack vector in a scenario. This perspective is far more important than the particular, preliminary results, which we obtain for it here. It is even more important than recoverable robustness itself. This perspective is a new way to understand optimization under imperfect knowledge for one of the most successful objects in applied mathematics, the linear program.

4.5.1 Introductory Examples

In this part we discuss three simple examples of LRPs with right-hand side disturbances. This discussion is meant to provide the geometric intuition behind the theorems on coincidental covering of LRPs in Subsection 4.5.3.

The Examples Two of the three LRPs arise from the simple robust timetabling problem on two simple graphs: The first graph is comprised of n connected components, each containing only a single arc. The second graph is a path with n arcs. The planning variable x_i represents the buffer time between the $(i-1)$-th event and the i-th event on the path. In the first graph x_i is the buffer time on the i-th of the n isolated arcs. Thus, nominal feasibility simply means $x \geq 0$. The resulting problems both have the following form:

$$\begin{aligned}
&\min c'x \\
&\quad \text{s.t.} \qquad x \geq 0 \\
&\qquad\qquad \forall b \in S\ \exists y \geq 0 : \\
&\qquad\qquad\qquad Ax + \hat{A}y \geq b \\
&\qquad\qquad\qquad d'y \leq D
\end{aligned}$$

For both problems we have $n = \hat{n} = m$ all dimensions equal and both planning matrices equal to the identity, $A^1 = A^2 := \mathbb{I}$. We set both planning and recovery costs equal to the all-one vector $c^1 = c^2 = d^1 = d^2 := (1, 1, \ldots, 1)'$, and also the recovery budget for both problems equal to $1 =: D^1 = D^2$. The difference is only in the recovery matrices $\hat{A}^1$ and $\hat{A}^2$. We suppress the variables for delay (recovery) at vertices without incoming edges, as they can always be chosen equal to zero. Thus, for the first problem we have $\hat{A}^1 := \mathbb{I}$ the identity, and for the second problem on the path we get the following:

$$\hat{A}^2 := \begin{pmatrix} 1 & 0 & 0 & \dots & & 0 \\ -1 & 1 & 0 & \dots & & \\ & & & \ddots & & \\ 0 & & & \dots & -1 & 1 \end{pmatrix}$$

Finally, define $S^k := \{b(\geq 0) : \|b\|_1 \leq k, \|b\|_\infty \leq 1\}$ and for simplicity let $n := 2$ and $k = 1$. In each worst case scenario, at most one arc is disturbed.

Because we assume hostile disturbances, we require $b \geq 0$.

What do recovery robust solutions look like? Where to put buffer times? In Problem 1 no buffer is needed. No matter which of the two arcs is disturbed, the total delay—i.e., recovery cost—will be 1, which is the recovery budget. In Problem 2 we have to distinguish whether the first or the second arc is affected. A disturbance on the second arc causes a total delay less or equal to 1 even without buffering. A disturbance on the first arc causes a delay of 2 without buffering. Thus, one must buffer in problem 2. Buffering either the first arc or the second with a unit buffer, would give a feasible solution. In fact, buffering the first arc with half a time unit is already sufficient for a feasible recovery.

What kind of coincidental covering can we expect in the second example? On a path early disturbances are more dangerous than late ones. A disturbance at one of the arcs in front can cause delay that propagates onto the many subsequent arcs. Whereas a disturbance on the last arc, can only affect that single arc. Thus, on a path we have to protect against influential disturbances. This might cause an over-protection in case less influential disturbances occur. So we expect coincidental covering. Indeed for Problem 2 we will a coincidental covering for every recovery robust solution, whereas for Problem 1 some solutions show no coincidental covering.

The problems both feature the non-negativity condition to the recovery vector $y \geq 0$. A negative recovery would mean that an event takes place earlier than planned in the schedule. This is forbidden by the timetabling condition. Also in many other conditions negative recovery is not allowed. We will see that the non-negativity condition for recovery causes a substantial structural difference to the standard case.

Finally, we add a third problem which again only differs in the recovery matrix: $\hat{A}^3 = (\hat{a}^3_{i,j} := |\hat{a}^2_{i,j}|)_{i,j \in \mathbb{R}^{m \times \hat{n}}}$. The recovery matrix of Problem 3 contains the absolute values of the matrix entries of Problem 2.

A Change in Perspective The feasibility condition is given by the inequality $Ax + \hat{A}y \geq b$. Re-write this as follows:

$$\hat{A}y \geq b - Ax$$

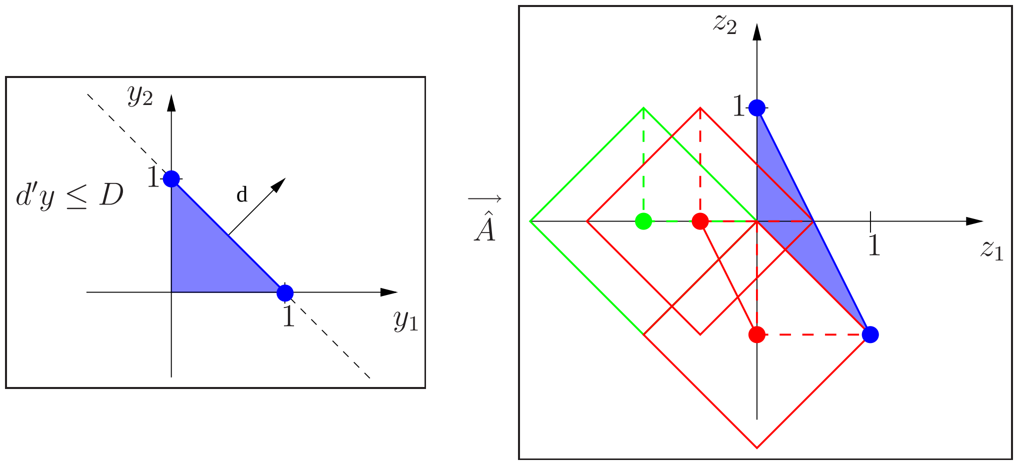


Figure 4.9: A simple example (Problem 2) in polyhedral perspective

In each row, on the right-hand side, we have the disturbance b minus the slack of that row in the planning problem $0 - Ax$. On the left, we find the image of the recovery vectors. Recovery means to find an admissible recovery vector that has an image, which is in each row greater or equal to the disturbance minus the slack. This motivates Figure 4.9, which we explain in detail now.

Three vector spaces are at stake: the n-dimension solution space of x, the $\hat{n}$-dimensional recovery space of y, and the m-dimensional row space—in which we denote vectors by z variables.

On the left of Figure 4.9 the vector space of the recovery variables y is depicted. The recovery cost d together with the budget D defines a hyperplane (the dotted line), below which those recovery vectors lie that respect the budget. The blue shaded area are the recoveries which also fulfill $y \geq 0$, thus these are the admissible recoveries. On the right of Figure 4.9 we depict $\hat{A}y$ for these admissible recovery vectors y in blue. We have for the non-trivial vertices of that simplex:

$$\hat{A}^2 \cdot \left\{ \left(\frac{D}{d_1}, 0 \right)', \left(0, \frac{D}{d_2} \right)' \right\} = \{(1, -1)', (0, 1)'\}.$$

Let us drop the index indicating the problem for a while as we concentrate on Problem 2.

Domination Rewriting the feasibility condition for the nominal solution x as above

$$\hat{A}y \geq b - Ax$$

allows the following interpretation of the right-hand part of 4.9. Consider the green point, which is $z := -A \cdot (1, 0)'$, the negative of the image of a solution

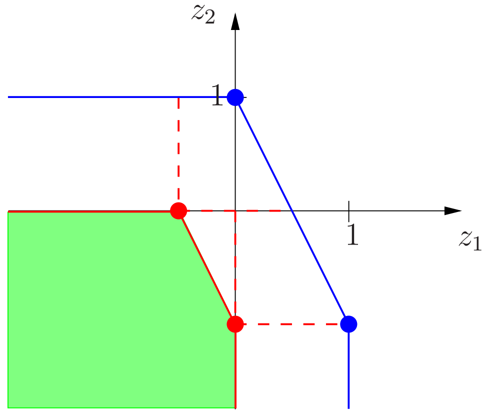


Figure 4.10: The negatives of the images of recovery robust solutions

we have seen to be recovery robust. The green diamond, is the set of points with distance to this solution less or equal to k $(= 1)$ in the 1-norm (and 1 in the infinity norm). For each point u in the green diamond exists a point w in the blue area which is greater or equal to u in every coordinate. We say the blue area dominates the green diamond. Denote the green diamond by U, then domination of the set U by the blue area means:

$$\forall z \in U : \exists y \geq 0, d'y \leq D : \hat{A}y \geq z.$$

By the definition of $U := \{z = b - Ax : \|b\|_1 \leq k, \|b\|_\infty \leq 1\}$ this is another way of saying that $x = (1, 0)$ is recovery robust.

The bold blue line shows $\hat{A} \cdot \{d'y = D\}$ which we call the recovery plane. The blue area dominates a set U if and only if the recovery plane dominates U.

The recovery plane also dominates the two red diamonds, which correspond to the solutions $x = (0, 1)$ and $x = (0.5, 0)$. The upper red solution cannot be moved further in positive z_1 direction, without some points being beyond the bold blue line. In other words, reducing the buffer on the first arc would leave scenarios for which the recovery exceeds the budget.

Non-negative Recovery Can we move the upper red compound further in positive z_2 direction when shifting in negative z_1 at the same time? If we move the solution as described, there is still a recovery vector y with recovery cost below D in every scenario. But some of the entries of that recovery vector need to be *negative*. If the problem requires $y \geq 0$ those would not be admissible recoveries, and thus the solution not recovery robust.

Note the structural difference between problems with non-negative recovery and the standard case. Considering the recovery space, in the non-negative case only the vectors in the blue area are admissible recoveries. In

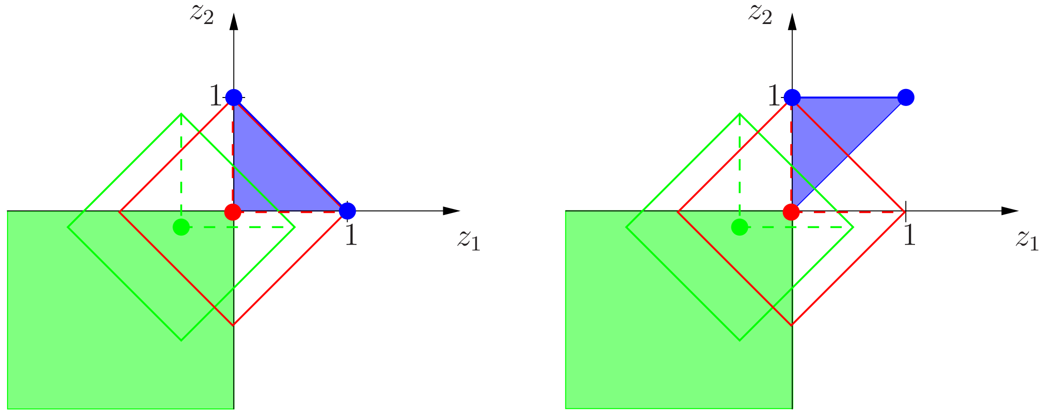


Figure 4.11: The row spaces of Problem 1 and 3

the general case the complete half space below the recovery plane is admissible.

Figure 4.10 shows the situation in the row space of Problem 2 with non-negative recovery. The negatives of the images of recovery robust solutions all lie in the green shaded area.

Vertices for Free In the proof of the Bertsimas-Sim bound the robust solution was constructed for a certain subset of the hypercube $[-1, 1]^n$, but turned out to cover for free a number of vertices of the hypercube outside that subset. The green and red diamonds in our pictures also lie in the hypercube of all scenarios respecting the interval bound. Are there any covered vertices of the hypercube outside the diamonds?

Of course, the lower vertices of the hypercube are covered. But, this is trivial, as it means that for example a driving activity took less time than technically assumed. It is not a very distinguished feature of a solution to remain feasible if things work better than expected.

The Bertsimas-Sim argument rests on an equivalent role for negative and positive disturbances. For right-hand side disturbance this equivalence is not given. Nevertheless, for certain problems we can identify vertices of the hypercube $[0, 1]^m$ which are covered coincidentally. Recall that these vertices are of particular importance, as they carry the whole probabilistic mass in worst-case distributions.

To this end compare Problem 1 and Problem 2. Both problems have the same recovery robust solutions, and a common, unique, optimal solution (depicted as a red vertex in Figure 4.11). Still, those problems behave quite different for scenarios outside S^1. Consider, the scenario spaces for higher k up to an m-dimensional hypercube $S^1, S^2, \ldots, S^{m-1}, S^m = C := \{\|b\|_\infty \leq 1, b \geq 0\}$. In our example S^2 is already the hypercube C.

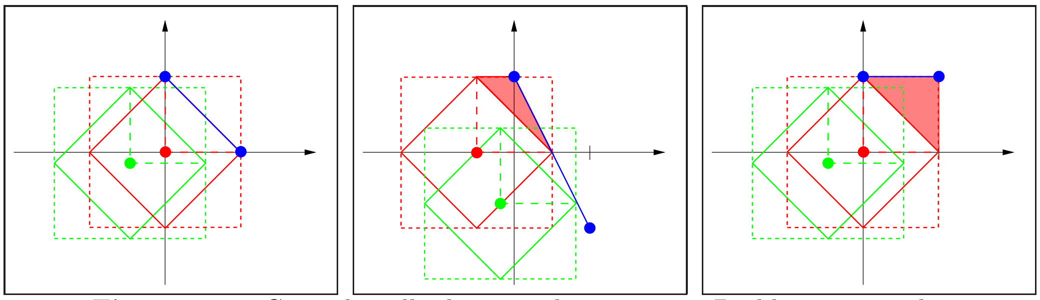


Figure 4.12: Coincidentally dominated scenarios in Problems 1, 2 and 3

Every recovery robust solution of Problem 3 over S^1 is also recoverable in all scenarios in C (cf. Figure 4.11). In contrast, the unique optimal recovery robust solution for Problem 1 with S^1 is not recoverable in any non-negative scenario of $C \setminus S^1$. Every solution to Problem 3 coincidentally covers 0-1-vectors, i.e., vertices of the hypercube $[0,1]^m$ outside the scenario set S^1. Whereas, Problem 1 seems to be sharp in the sense that the solution may not cover any other scenario than those it has been constructed for.

Further Coincidental Covering For Problem 3 we could find covered 0-1-vectors outside S^1. Problem 1 behaves sharp. The sharpness seems to depend on the angle between the recovery plane and the dominating surface of S^1. Problem 2 also has a non-vanishing angle between these two objects. But, there are solution to Problem 2 that do not cover any 0-1-vector outside S^1. Still, the interpretation of Figure 4.9 suggests that also for Problem 2 every solution that is recoverable in all scenarios of S^1 is also recoverable in *some* non-negative scenarios of $C \setminus S^1$.

In the middle of Figure 4.12 we show the row space of Problem 2. For the red solution it depicts as a red shaded area a set of scenarios outside S^1, which is coincidentally covered. The same area in the environment of the green solution is also coincidentally covered. It seems to be an invariant set of scenarios for all recovery robust solutions.

To summarize Figure 4.12 shows three types of situation:

- In Problem 1 (left) no coincidental covering might happen.
- In Problem 2 (middle) coincidental covering is guaranteed.
- For Problem 3 (right) the coincidental covering necessarily also affects 0-1-vectors outside S^1.

Strict Robustness Consider the situation for strict robustness. As we require $y \geq 0$ and have $d \geq 0$ we can enforce strict robustness by setting

$D = 0$. Thus, there is only one admissible recovery, namely, the zero vector $y = \mathbf{0}$. Its image $\hat{A}\mathbf{0}$ is a point, notably, the zero vector in the row space.

A solution x is recovery robust over S^k if and only if $b - Ax$ is dominated by $\mathbf{0}$ for all $b \in S^k$. The subset dominated by the origin is an orthant, namely

$$\{x \in \mathbb{R}^m : x_i \leq 0 \,\forall 1 \leq i \leq m\}.$$

From the geometric intuition it seems clear that this for any $k > 0$ requires the whole hypercube S^m to be covered as well. Indeed, this intuition will be justified by the more general Theorem 4.22.

This is a different way to formulate an observation we pointed out earlier: If one requires strict robustness against a certain disturbance in one unknown row, one has to protect against this disturbance in all rows. Strict robustness has no sight for vertically integrated limits to the scenario set.

Summary Requiring recoverable robustness against S^k enforces recoverable robustness for scenarios outside S^k, too. This effect becomes apparent in considering the relative position of the recovery plane and the negative of the solution's slack vector surrounded by the scenario set. The key phenomenon is the domination of the latter by the recovery plane. The coincidental covering is controlled by the shape of the scenario set S^k and the image of the admissible recoveries under the recovery matrix $\hat{A}$. The original optimization problem plays a minor role in this perspective.

The dominated area of this image is substantially different, if recovery must be non-negative. For the standard case, the dominated area seems to be a half space. In the non-negative case, it is a more complex structure. It will turn out to be the Minkowski-sum of a polytope and a cone.

Finally, we could observe a difference between the standard cases like Problem 1 and 2, and those situations in which 0-1-vectors outside the scenario set S^k were coincidentally covered, like Problem 3 (or strict robustness).

These observations may depend heavily on peculiarities of $\mathbb{R}^2$ and the chosen examples. In the next section we will give a general theorem for each of the observed phenomena.

In terms of the timetabling application coincidental covering reads as follows: The guarantee that we will not exceed a total delay of D in any scenario, which has at most k disturbances of size at most Δ, enforces a timetable that will also cope with some scenarios of higher total disturbance. For example, early delays—i.e., delays at edges with many successors—cause more problems than late delays. A sufficient buffering against limited early delays is also sufficient against higher, or more delays in the later parts of the network. Quantifying the shaded region in Figure 4.12 means to quantify a priori this property of a network buffering. The effect clearly depends on the networks topology: A path has early and late arcs, whereas in Problem 1 all arcs are similar. Correspondingly, Problem 2 shows coincidental covering, whereas for Problem 1 there are solution without coincidental covering. To analyze this property of the network it suffices to analyze the image of the admissible recoveries under the recovery matrix $\hat{A}$.

4.5.2 Preliminaries

We want to distill some of what we have observed in the example into proven theory.

Due to the type of limits for the disturbances, the choice of the basis for the vector spaces is not arbitrary, but prescribed by the modeled application. Therefore, we consider all vector spaces with their fixed basis. We denote the i-th basis vector by $e^i, e^i_i = 1, e^i_j = 0$ for all $j \neq i$. By $[\frac{D}{d_i}]$ we denote the set $\hat{n}$ vectors $v^i \in \mathbb{R}^{\hat{n}}$ $(0 \leq i \leq \hat{n})$ with the i-th component equal to $v^i_i = \frac{D}{d_i}$ and zero else. These together with the origin are the edges of the simplex $\{0 \leq y \in \mathbb{R}^{\hat{n}} : d'y \leq D\}$. As a slightly unusual notation we use AS for a matrix A and a subset S of a suitable vector space, to express the image of the set S under the linear mapping represented by the matrix A for the canonical basis. Note that for vectors a, b we mean by $a < b$ that for *every* i the entry b_i is strictly larger than the entry a_i.

We start by some preliminary observations on dominance.

Definition 4.16. *Let $U, L \subseteq \mathbb{R}^n$ be subsets of a vector space over the reals. We say U* dominates *L, write $L \prec U$, if*

$$\forall l \in L \, \exists u \in U : u - l \geq 0.$$

From the definition we get immediately the following observation.

Observation 4.17. *The dominance relation is reflexive and transitive.*

Still, it is not a partial order as it fails antisymmetry even in dimension 1: $\{0, 1\} \prec \{1\} \wedge \{1\} \prec \{0, 1\}$. Restricting dominance to sets with one element antisymmetry is fulfilled. Thus the set of singletons forms a lattice. In fact for singletons the $\prec$-relation equals the $\leq$-relation. In dimension greater than 1 there are incomparable pairs of both sets and singletons. For singletons we use $a \prec b$ for $\{a\} \prec \{b\}$.

Observation 4.18. *Dominance is invariant under translation, i.e., for all $a \in \mathbb{R}^n$ and $U, L \subseteq \mathbb{R}^n$ we have $U \prec L \Leftrightarrow U + a \prec L + a$.*

The following observation is helpful when dealing with the typical scenario sets $S^k := S^k(b^*) := \{b^* + z : 0 \leq z_i \leq 1 \wedge \sum z_i \leq k\}$. (In fact the observation remains true, if we define $S^k(b^*) := \{b^* + z : -1 \leq z_i \leq 1 \wedge \sum z_i \leq k\}$.)

Observation 4.19. *For each $k \geq 1$ we have $S^k \prec \overline{S^k} := \{b^* + z : 0 \leq z_i \leq 1 \wedge \sum z_i = k\}$, i.e., the set S^k has a dominating facet $\overline{S^k}$.*

Invariance under translation allows us to consider $S^k(0)$ instead of $S^k(b^*) - Ax$, and by transitivity we can confine our domination considerations to $\overline{S^k}$.

Definition 4.20. *For a vector z define U_z as the set of all vectors u, such that $z \prec u$, and L_z as the set of all vectors ℓ, such that $\ell \prec z$, i.e.:*

$$U_z = \{d \in \mathbb{R} : \forall i : d_i \geq z_i\}$$

and

$$L_z = \{d \in \mathbb{R} : \forall i : d_i \leq z_i\}.$$

For a set D the dominated set is defined by

$$L_D := \bigcup_{d \in D} L_d$$

and the dominating set by

$$U_D := \bigcup_{d \in D} U_d.$$

Note that $U_z \cap L_z = \{z\}$. For a convex set D the dominated set L_D is the Minkowski sum $D + O^-$, with $O^- = \mathbb{R}^m_{\leq 0}$ the orthant of vectors with all entries non-positive, and $U_D = D + O$, with $O = \mathbb{R}^m_{\geq 0}$ the positive orthant. As the Minkowski-sum of convex sets is convex we can make a very important observation:

Observation 4.21. *The set dominated by a convex set is convex.*

4.5.3 Standard Linear Recovery Robust Programs

Recall the definition of the principal object of our considerations, the Linear Recovery Robust Program:

Definition 4.5 Let A_0 be an $m \times n$-matrix called the *nominal matrix*, b_0 be an m-dimensional vector called the *nominal right-hand side*, c be an n-dimensional vector called the *nominal cost vector*, $\hat{A}$ be an $m \times \hat{n}$-matrix called the *recovery matrix*, d be an $\hat{n}$-dimensional vector called the *recovery cost vector*, and D be a non-negative number called the *recovery budget*. Further let S be a closed set of pairs of $m \times n$-matrices and m-dimensional vectors containing (A_0, b_0), called the *scenario set*. Then the following optimization problem is called a *Linear Recovery Robust Program (LRP)* over S:

$$\begin{aligned}
&\min_x c'x \\
&\quad \text{s.t.} \qquad A^0 x \geq b^0 \\
&\qquad \forall (A,b) \in S \; \exists y \in \mathbb{R}^{\hat{n}} : \\
&\qquad\qquad Ax + \hat{A}y \geq b \\
&\qquad\qquad d'y \leq D
\end{aligned}$$

We use $R := \hat{A}\{y \in \mathbb{R}^{\hat{n}} : d'y \leq D\}$ and $\bar{R} := \text{span}(\hat{A}[\frac{D}{d_i}])$. We call $\bar{R}$ the *recovery plane*. Note that $\text{span}(\hat{A}[\frac{D}{d_i}]) = A\{y \in R^{\hat{n}} : d'y = D\}$.

For an m-dimensional LRP define

$$T = \{e^i, 1 \leq i \leq m : (r^1 - r^2)'e^i = 0 \, \forall r^1, r^2 \in \bar{R}\}$$

as the subset of basis vectors in the row space orthogonal to $\bar{R}$. Further, set $\tau := |T|$. With this notation we give the following theorem.

Theorem 4.22. *Let x^* be a recovery robust solution to an m-dimensional LRP over S^k with $k \geq 1$. Then for any $k < \ell \leq m$, the solution x^* coincidentally covers at least*

$$\sum_{i=l-k}^{\min(\tau,l)} \binom{\tau}{i}\binom{m-\tau}{l-i}$$

0-1-*vectors with exactly l positive entries.*

Proof. For any vector v we say an index i or its entry v_i in a vector $v \in \mathbb{R}^m$ is *represented in T*, if $e^i \in T$.

We show for every 0-1-vector with exactly l positive entries that it is dominated by a vector in R, if at least $l - k$ of these entries are represented in T. Then we count these vectors.

Say z^ℓ is such a vector with ℓ entries equal to 1. There is a vector $z^k \in S^k$ such that $z^\ell - z^k$ is a 0-1-vector in the span of T. As x^* is recovery robust, and by $k \geq 1$ each basis vector e^j is in the scenario set S^k, we have:

$$\exists \hat{A}y^k = r^k \in R : z^k \prec r^k$$

$$\exists \hat{A}y^j = r^j \in R : e^j \prec r^j$$

Define $\lambda := \frac{d'y^j}{d'y^k}$, and $y^* := \lambda y^k$. Finally, set $\bar{y} := y^* - y^j$. Note $d'\bar{y} = 0$ and thereby $(\hat{A}\bar{y})'e^j = 0$ as e^j is orthogonal to $\bar{R}$. This gives $e^j \prec \hat{A}(y^j + \bar{y}) = \hat{A}y^*$.

Further, there is a $\lambda^* \geq 0$ such that $\lambda^* y^k = y^k - y^*$. As $z^k \geq 0$ also its dominator $r^k \geq 0$, and thereby $\lambda^* r^k \geq 0$. We can conclude:

$$e^j \prec \hat{A}y^* + \lambda^* r^k = r^k$$

As r^k dominates z^k and e^j it also dominates their sum.

The argument holds for every positive entry of $z^\ell - z^k$ and fixed r^k. Hence, r^k dominates z^ℓ, too.

To count the number of different 0-1-vectors with exactly l positive entries and at least $l-k$ entries represented in T, divide the set of all basis vectors in those τ represented in T and the $m-\tau$ others. To construct one of the desired vectors unequivocally we fix the number of positive entries represented in T by the counting parameter i. Thus, we choose i in τ and $\ell - i$ in $m - \tau$. This gives the counting formula. □

The proof works as well for the case, when the recovery is restricted to be non-negative, i.e., when $y \geq 0$ is required for every scenario.

Corollary 4.23. *Let x^* be a recovery robust solution to an m-dimensional LRP over S^k with $k \geq 1$, and additional requirement $y \geq 0$. Then for any $k < \ell \leq m$, the solution x^* covers at least*

$$\sum_{i=l-k}^{\min(\tau,l)} \binom{\tau}{i}\binom{m-\tau}{l-i}$$

0-1-*vectors with exactly l positive entries.*

A special case of the non-negative recovery is treated in the next corollary. It contains the case of strict robustness.

Corollary 4.24 (Strict Robustness)**.** *Let x^* be a recovery robust solution to an m-dimensional LRP over S^k with $k \geq 1$, and additional requirement $y \geq 0$ and $D = 0$. Then x^* covers the hypercube S^m.*

Note that the preceeding lemma models strict robustness if $d > 0$.

The above theorems motivate to restrict our considerations to LRPs, which fulfill certain further conditions. The first condition avoids the peculiarity that cheaper recovery vectors are strictly more useful in recovery. The second one is similar to that of *complete recourse* in stochastic programming.

Definition 4.25. *Let $(A_0, b_0, c, \hat{A}, d, D)$ define an LRP over some closed set S.*

- *The LRP is said to have* cost-aware recovery *if the recovery plane $\bar{R}$ is a minimal dominating subset of R.*
- *The LRP is said to have* complete recovery*, if the* $\dim(R) = m$.
- *An LRP with complete and cost-aware recovery is called a* Standard Linear Recovery Robust Program (SLRP).

We have seen in the motivating example, even if no 0-1-vector outside S^k is covered coincidentally, a large set of other scenarios outside S^k is covered coincidentally. Assume we are given a σ-algebra and a probability measure μ for the set of all scenarios in S^k. Then for an SLRP we can calculate a lower bound for the probability mass of the scenarios between $\overline{S^k}$ and the recovery plane.

Algorithm 2: VOL(ν, k, μ)

```
Compute the maximal edge $e$ of $\overline{S^k}$ for the outer normal vector $\nu$ to $\bar{R}$
as objective function.
Compute a translation of $\bar{R}$ by a vector $\tau$, such that $\bar{R} + \tau$ contains $e$.
Compute $V_1$, the measure with respect to $\mu$ of the intersection
$S^m \cap H_{\bar{R}+\tau}$ of the hypercube with the half-space below $\bar{R} + \tau$.
Compute $V_2$, the measure of $S^k$ with respect to $\mu$.
Return $V_1 - V_2$.
```

Theorem 4.26. *Let $(A_0, b_0, c, \hat{A}, d, D)$ define an SLRP over S^k, and x^* be a recovery robust solution to it. Let ν be the outer normal vector of $\bar{R}$ and μ some probability measure for the space of right-hand side disturbances b with a suitable σ-algebra. Then the output $VOL(\nu, k, \mu)$ of Algorithm 2 gives a lower bound on the probability under μ that x^* is recovery robust.*

Proof. Any vector in the area measured by the algorithm has smaller distance to the recovery plane, than the vertex of S^k which is maximal with respect to ν. As that maximal vertex is covered by recoverable robustness of x^*, the maximal vertex has at most distance 0 from the recovery plane. Therefore, all vectors in the area measured by the algorithm have non-positive distance to the recovery plane, i.e., they are covered by $\bar{R}$. □

4.5.4 Jealous Linear Recovery Robust Problems

The examples motivate to distinguish between SLRPs where the recovery variables can be negative, and those where they are a priori required to be non-negative. We will call the later *jealous*, because figuratively a row jealously keeps its slack, whereas negative recovery allows to transfer planned slack from one row to another. Jealous SLRPs play an important role in practice. The robust timetabling problem is jealous, because of the timetabling condition: No train is allowed to run ahead of schedule, i.e., produce a negative recovery.

Definition 4.27. *Let $(A_0, b_0, c, \hat{A}, d, D)$ define an SLRP over a closed scenario set S. Then the following optimization problem is called a* Jealous Linear Recovery Robust Program (JLRP) *over S:*

$$\begin{aligned} &\min_x c'x \\ &\quad s.t. \qquad\qquad A^0x \geq b^0 \\ &\qquad\qquad \forall (A,b) \in S\ \exists 0 \leq y \in \mathbb{R}^{\hat{n}} : \\ &\qquad\qquad\qquad\qquad\qquad\qquad Ax + \hat{A}y \geq b \\ &\qquad\qquad\qquad\qquad\qquad\qquad d'y \leq D \end{aligned}$$

At first sight, jealousy seems enforcable for an SLRP without explicitly requiring $y \geq 0$, by the choice of the planning and the recovery matrix and the scenario set. At second sight, a subtle difference remains. To impose $y \geq 0$ indirectly, one has to enlarge the dimension of the row space. This would affect the completeness of recovery. Analyzing coincidental covering of JLRPs is a little less trivial than for SLRPs. In an SLRP the recoverable area in the row space is the half space below the recovery plane $\bar{R}$. For JLRPs the role of $\bar{R}$ is played by the *recovery polytope* $\dot{R} := \hat{A}\{y \geq 0 : d'y = D\}$.

In order to analyze coincidental covering by JLRPs we make some basic observations:

- The recovery polytope $\dot{R}$ is a subset of the recovery plane $\bar{R}$ by definition. The area covered by $\dot{R}$ is the Minkowski-sum of that polytope with $O^- = \mathbb{R}^m_{\leq 0}$ and therefore a convex set.

- The position of $\bar{R}$ or $\dot{R}$ in the row space is fixed, and the negative of the slack of a recovery robust solution must be in its domination area for all scenarios. It is more convenient to think in the sequel of $b^* - Ax = 0$ as fixed and the recovery polytope as moving. Obviously, all that matters for dominance is the relative position of the recovery polytope and $b^* - Ax$ surrounded by the disturbance set S^k. The question is the following: If $S^k + (b^* - Ax)$ lie in the area dominated by $\dot{R}$, what else must be in that area?

- Assuming $b^* - Ax$ fixed to the origin the following definition makes sense. Consider the family of all hyperplanes parallel to the recovery plane. Define $\bar{R}_0$ as the lowest of those hyperplanes, which still dominates S^k. (This is the one used in Algorithm 2.) The recovery polytope lies in one of those hyperplanes. As $\dot{R}$ dominates S^k the hyperplane containing $\dot{R}$ also dominates S^k. Therefore, the recovery polytope lies in a hyperplane above $\bar{R}_0$.

- For each vector $s \in S^k$ there is a vector $d_s \in \dot{R}$ with $s \prec d_s$. Consider the line segment $\ell := [s, d_s]$. All vectors in ℓ dominate s. The covered vector s is below or in $\bar{R}_0$, but $d_s \in \dot{R}$ and therefore above or in $\bar{R}_0$. Thereby, ℓ must intersect $\bar{R}_0$. This intersection is a dominating point $d_s^0 \in \bar{R}_0$ for s in the lowest dominating hyperplane.

- Next, we ask for the set of vectors in $\bar{R}_0$ which are candidates for d_s^0, i.e., which could dominate s. Recall that $O = \mathbb{R}^m_{\geq 0}$ denotes the positive orthant. For each basis vector e_i we have $\nu' e_i \neq 0$, because of cost-aware recovery. Therefore, $\bar{R}_0 \cap (O + s)$ is an $m-1$ dimensional simplex. We can find its m vertices by calculating by calculating the intersection point $\{s + \lambda e_i : \lambda \geq 0\} \cap \bar{R}_0$. Denote this finite set of vertices by $C^* = C_s^*$, and their convex hull by $C := C_s := \operatorname{conv} C^*$. Indeed, the simplex C is the set of all dominating points for s in $\bar{R}_0$. Thus, d_s^0 must lie in C.

- As the set of dominated vectors is convex, we know that $\operatorname{conv}(d_s^0, S^k)$ is dominated for every $s \in S^k$. We do not know which vector in C is d_s^0. But, it is sure that for any $s \in S^k$ all vectors in $\bigcap_{c \in C_s} \operatorname{conv}(c, S^k)$ are dominated by $\dot{R}$. So we know that

$$\bigcup_{s \in S^k} \bigcap_{c \in C_s} \operatorname{conv}(c, S^k)$$

is dominated by $\dot{R}$ in all feasible solutions. The following lemma allows to calculate $\bigcap_{c \in C_s} \operatorname{conv}(c, S^k)$ by finitely many intersections:

Lemma 4.28. *For C^* and P^* two finite sets in $\mathbb{R}^n$:*

$$\bigcap_{c \in C^*} \operatorname{conv}(c, P^*) = \bigcap_{c \in \operatorname{conv} C^*} \operatorname{conv}(c, P^*).$$

Proof. Define $C := \operatorname{conv}(C^*)$ and $P := \operatorname{conv}(P^*)$. One inclusion is trivial. We have to show for any $c \in C$ and any x, which is in $\operatorname{conv}(c_i, P)$ for all $c_i \in C^*$ that x can be expressed as a convex combination of P^* and c. Fix an x such that

$$\forall c_i \in C^* \; \exists p_i \in P, 0 \leq \mu_i \leq 1 : x = \mu_i c_i + (1 - \mu_i) p_i.$$

and an arbitrary $c \in C$, i.e., there are non-negative coefficients γ_i such that:

$$c = \sum_{i:c_i \in C^*} \gamma_i c_i \text{ and } \sum_{i:c_i \in C^*} \gamma_i = 1.$$

Define $\tilde{\xi}_i := \frac{\gamma_i}{\mu_i}$, $\Xi = \sum_{i:c_i \in C^*} \tilde{\xi}_i$, and $\xi_i := \frac{\tilde{\xi}_i}{\Xi}$. This means that $\sum_{i:c_i \in C^*} \xi_i = 1$. Therefore, we can write:

$$\begin{aligned} x &= \sum_{i:c_i \in C^*} \xi_i [\mu_i c_i + (1 - \mu_i) p_i] \\ &= \sum_{i:c_i \in C^*} \frac{1}{\Xi} \gamma_i c_i + \sum_{i:c_i \in C^*} \xi_i (1 - \mu_i) p_i \\ &= \frac{1}{\Xi} c + \sum_{i:c_i \in C^*} \xi_i (1 - \mu_i) p_i \end{aligned}$$

To see that $x \in \operatorname{conv}(c, \{p_i\}_{i:c_i \in C^*})$ it suffices to show that the coefficients in the last expression sum up to 1.

$$\begin{aligned} \frac{1}{\Xi} + \sum_{i:c_i \in C^*} \xi_i (1 - \mu_i) &= \frac{1}{\Xi} + \sum_{i:c_i \in C^*} \frac{\tilde{\xi}_i}{\Xi} (1 - \mu_i) \\ &= \frac{1}{\Xi} \left(1 + \sum_{i:c_i \in C^*} \frac{\gamma_i}{\mu_i} (1 - \mu_i) \right) \\ &= \frac{1}{\Xi} \left(1 + \sum_{i:c_i \in C^*} \frac{\gamma_i}{\mu_i} - \sum_{i:c_i \in C^*} \gamma_i \right) \\ &= \frac{1}{\Xi} (1 + \Xi - 1) = 1 \end{aligned}$$

□

Let S^* be the vertices of $\overline{S^k}$, i.e., the 0-1-vectors with exactly k positive entries. Give indices to the elements of this finite set from 1 to $|S^*| = r$.

Using again that the set dominated by the convex set $\dot{R}$ is itself convex we can apply Lemma 4.28 to conclude further that

$$\operatorname{conv}\left(\bigcup_{s\in S^k}\bigcap_{c\in C_s^*}\operatorname{conv}(c, S^k)\right) = \operatorname{conv}\left(S^k, \bigcap_{C_0\in\left\{C=\{c_1,c_2,\ldots,c_r\}:c_i\in C_{s_i}^*\right\}}\operatorname{conv} C_0\right)$$

is dominated.

Thus, we arrive at the following theorem describing for any JLRP a set of scenarios that is coincidentally covered by every recovery robust solution.

Theorem 4.29. *Let $(A_0, b_0, c, \hat{A}, d, D)$ define a JLRP over S^k. Then any recovery robust solution covers the following subset of scenarios coincidentally:*

$$\operatorname{conv}\left(\overline{S^k}, \bigcap_{C_0\in\left\{C=\{c_1,c_2,\ldots,c_r\}:c_i\in C_{s_i}^*\right\}}\operatorname{conv} C_0\right)\setminus\overline{S^k}$$

4.5.5 Disturbances in the Planning Matrix

In this part we discuss recoverable robustness for disturbances in the entries of the planning matrix A. The results of Bertsimas and Sim [22] accomplish the vertical integration for this setting in two respects: Firstly, they show that the (strict) robust counterpart to a linear program is still solvable in polynomial time. Secondly, they give a stochastic bound on the probability that a solution of their model will be feasible for scenarios that violate the Γ bound—in our terminology, scenarios outside S^k.

For the recovery robust setting we can achieve the same result with respect to complexity: our model stays tractable. The recovery robust approach is a lot more flexible in the way by which it achieves robustness. This in turn causes problems in deriving a stochastic bound. To be sure, the stochastic bound of Bertsimas and Sim applies in the recoverable robust setting, too. But, as in the right-hand side case, we are interested in hostile disturbances. We will explain why in this case an a priori bound on coincidental covering cannot be achieved by techniques similar to those proposed above. The recovery robust solutions can adopt themselves so perfectly to the given scenario set and the recovery matrix that they rather spare first stage costs than create coincidental covering. These solutions possess less secondary virtues (coincidental covering), because they better fulfill their primary assignment, namely, to minimize the objective function while covering the scenarios in S.

For a system of inequalities and disturbances only in the right-hand side it is a priori clear for each inequality, whether a positive or a negative disturbance is harmful. Assuming w.l.o.g. the form $Ax \geq b$ for a minimization problem, the hostile problem is confined to positive disturbances, i.e., $b^s = b^* + z^s$ with $0 \leq z^s \leq \Delta_\infty$. In case the planning solution $x \leq 0$ must be non-negative, also disturbances in the planning matrix A have a single hostile direction. Every positive disturbance works in favor of feasibility. Therefore, a stochastic claim about reliability resting on assumptions for *lower* bounds on the probability of positive disturbances is at least not surprising. The fact that the computed solution is still feasible, in case things turn out better than expected, will not impress a practitioner. So we request $x \geq 0$ and assume all disturbances in the planning matrix to be hostile, i.e., negative.

In case x is not required to be non-negative by the application, we use a substitution as in the Soyster model (cf. Problem 4.30).

To combine horizontal and vertical integration of the protection we limit the disturbances row-wise by $\bar{k}$ and in total by k. For convenience we split the planning matrix $A = A^* - \tilde{A}$ into the matrix A^* of the reference scenario minus the *disturbance matrix* $\tilde{A}$, for which we require every entry to lie between 0 and $\Delta_\infty = 1$. This means, we define the set of all likely scenarios $S^{\bar{k},k}$ via the set of disturbance matrices

$$\tilde{S} := \{A \in \mathbb{R}^{m,n} : \sum_j a_{ij} \leq \bar{k}, \sum_{ij} a_{ij} \leq k, 0 \leq a_{ij} \leq \Delta_\infty\}.$$

Assume k and $\bar{k}$ integer and $\Delta_\infty = 1$. The fractional cases can be handled in a similar way. But we confine this presentation to the integral case for clarity.

All together, the central object of this section are SLRPs of the following form:

$$\begin{aligned} \min c'x & \\ \text{s.t.} \quad & A^0 x \geq b^0 \\ & \forall A \in \{A^* - \tilde{A}, \tilde{A} \in \tilde{S}\} \exists y : \\ & Ax + \hat{A}y \geq b \\ & x \geq 0 \end{aligned}$$

with

$$\tilde{S} := \{A \in \mathbb{R}^{m,n} : \sum_j a_{ij} \leq \bar{k}, \sum_{ij} a_{ij} \leq k, 0 \leq a_{ij} \leq 1\}$$

The Adversary's Choice In case of right-hand side disturbances, which we consider in the previous parts, the adversary can influence directly the situation in the row space. A disturbance of size δ in the i-th component of the right-hand side causes directly a change of δ in the direction of the i-th basis vector. In case the adversary influences the disturbance matrix, its power is indirect. A change in a_{ij} influences in the direction of the i-basis vector to an extend that depends on the value of the j-th component x_j of the planning solution. Therefore, it makes sense for disturbed matrices to distinguish between the disturbance and the *aberration*, i.e., the change in the row space caused by the disturbance. The image of a planning solution under the reference scenario's matrix A^* is the origin of an *aberration polytope*[2] containing all vectors in the row space to which the adversary can defer the original image of the solution. These are the objects that receive our attention in the sequel.

In particular we are interested in those vectors of the aberration polytope, that correspond to worst case scenarios. These are surprisingly easy to find. For a fixed solution x the adversary aims to choose a matrix $\tilde{A} \in \tilde{S}$ such that

$$\forall y : \hat{A}y \geq b - Ax + \tilde{A}x \Rightarrow d'y > D.$$

In other words, the adversary maximizes $(\tilde{A}x)'\nu$ over all matrices in $\tilde{S}$, where ν is the outer normal vector of $\bar{R}$. We can describe these maximal solutions quite well. The key observation is that for a fixed tuple of row sums $(a_i)_i$ of the disturbance matrix $\tilde{A} = (a_{ij})_{i,j}$ the maximal scalar product of $\tilde{A}x$ and ν can be calculated in a straight forward way.

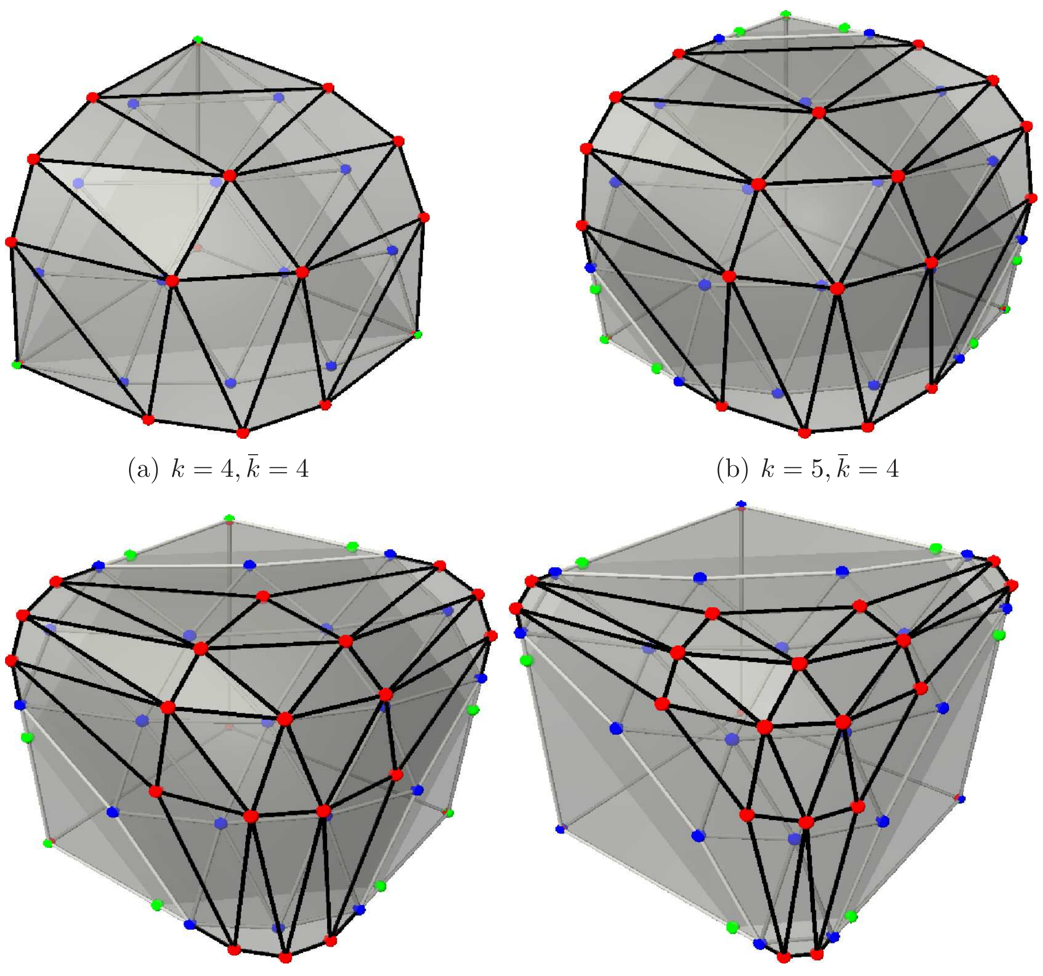
Observation 4.30. *Let $a_i := \sum a_{ij}$ be the sum of the i-th row of the disturbance matrix.*

1. *For fixed x and fixed values $a_i, 1 \leq i \leq m$, a disturbance matrix maximizing $(\tilde{A}x)'\nu$ can be constructed in the following way: In the i-th row set the coefficients for the $\lfloor a_i \rfloor$ largest entries of x equal to 1, and that of the $(\lfloor a_i \rfloor + 1)$-th largest entry of x equal to $a_i - \lfloor a_i \rfloor$, and the rest zero.*

2. *Let the sums a_i again be variables within the limits $\sum_{0\leq i\leq m} a_i \leq k$ and $0 \leq a_i \leq \bar{k}$. The products $\tilde{A}x$ of the disturbance matrices resulting from the above construction, and the fixed solution x lie in a polytope spanned by those products, which stem from tuples $(a_i)_i \in \mathbb{N}^m$ of* integral *row sums*[3].

[2]At this point it may not be obvious that the set of possible aberrations is actually a polytope for $S_{\bar{k},k}$. But this will become clear soon.

[3]At this point we exploit the integrality of $\bar{k}$ and k. For fractional values one would



(a) $k = 4, \bar{k} = 4$ (b) $k = 5, \bar{k} = 4$

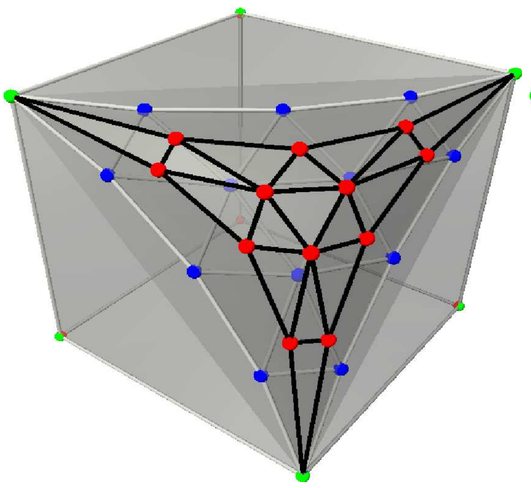
(c) $k = 6, \bar{k} = 4$ (d) $k = 7, \bar{k} = 4$

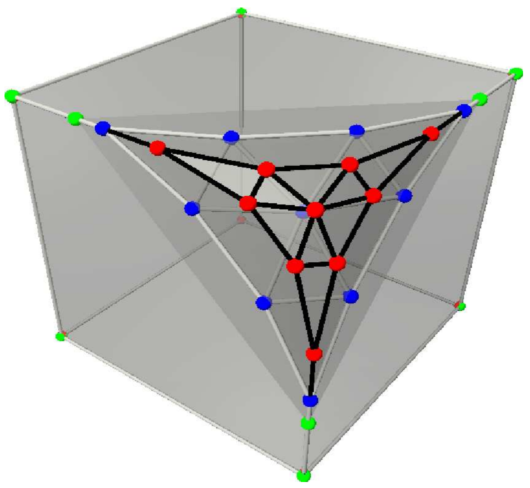
Figure 4.13: Aberration polytopes for solutions with maximal entries $(8, 4, 2, 1)$ (red vertices), $(5.25, 4.25, 3.25, 2.25)$ (green), and $(3.75, 3.75, 3.75, 3.75)$ (solid) for $4 \leq k \leq 7$.

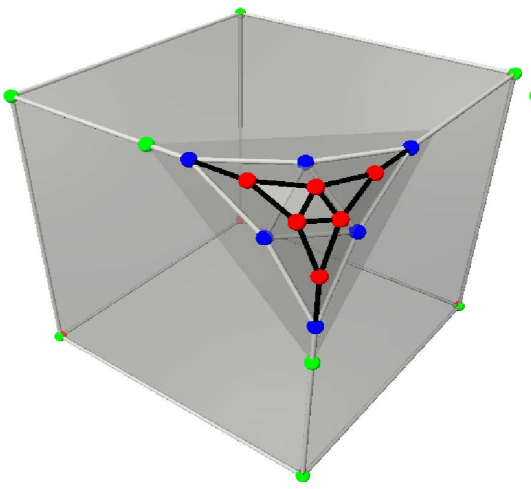
3. *Enumerate all m-tuples $(a_i)_i \in \mathbb{N}^m$ of integers with $0 \leq a_i \leq \bar{k}$ and $\sum a_i = k$. This gives all row sums, which characterize the disturbance matrices of a possible worst case scenario. The corresponding aberrations form finite superset for the vertices of the aberration polytope. Note that this enumeration is independent of the solution x.*

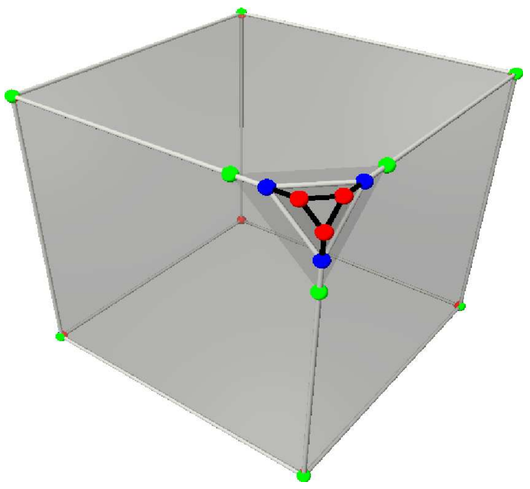
An example will make this observation more transparent. Use $\bar{x}_i$ to denote the i-th greatest entry of a solution x. Let $\bar{k} = 4$ and $\bar{x}_1 = 8, \bar{x}_2 = 4, \bar{x}_3 = 2$, and $\bar{x}_4 = 1$. By, the above observations we need not consider $\bar{x}_i$

have to keep track of the fractional rest, which is possible, but distracts the attention from the important insights.


(a) $k = 8, \bar{k} = 4$


(b) $k = 9, \bar{k} = 4$


(c) $k = 10, \bar{k} = 4$


(d) $k = 11, \bar{k} = 4$

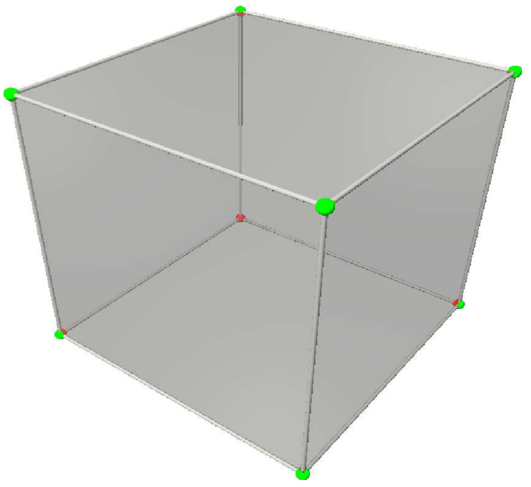
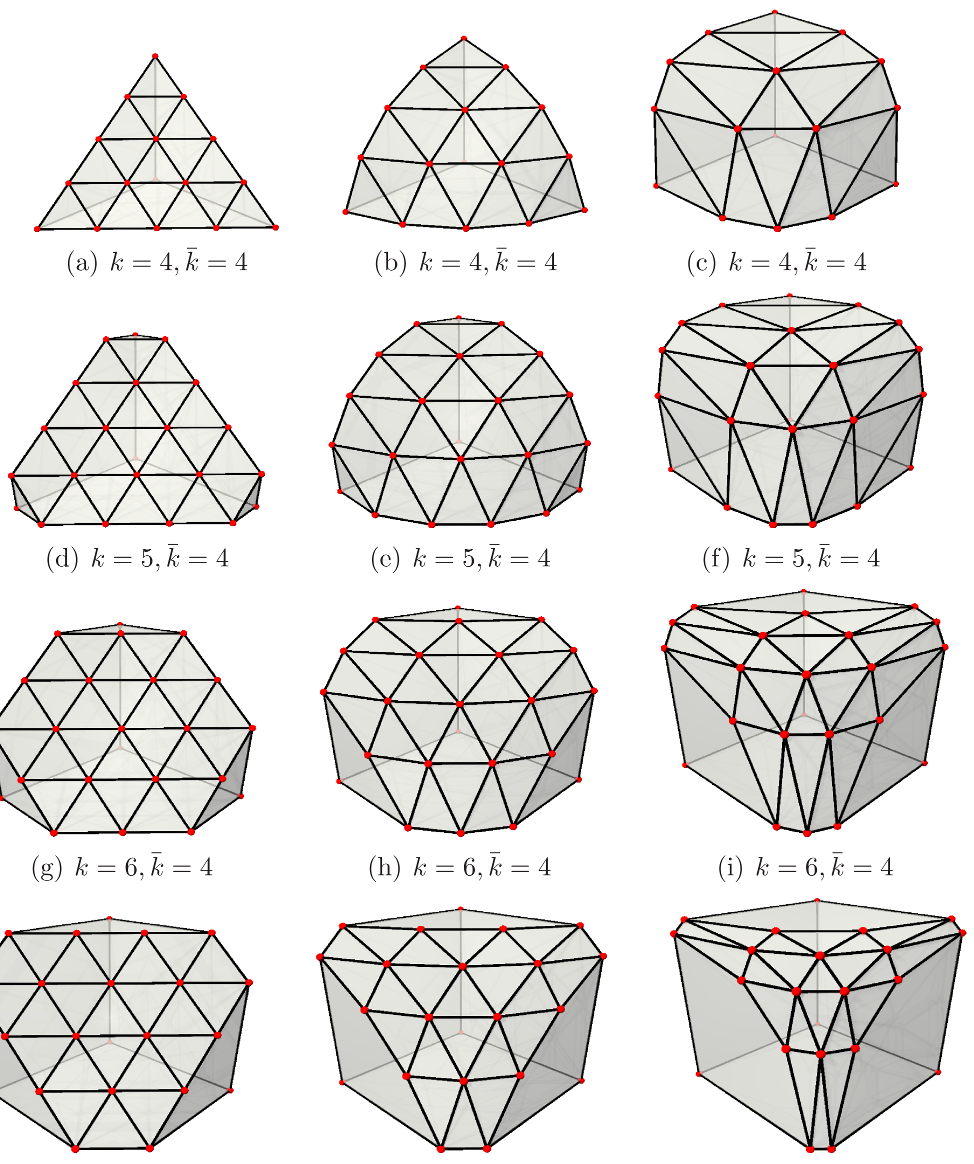

(e) $k = 12, \bar{k} = 4$

Figure 4.14: Aberration polytopes for solutions with maximal entries $(8, 4, 2, 1)$ (red vertices), $(5.25, 4.25, 3.25, 2.25)$ (green), and $(3.75, 3.75, 3.75, 3.75)$ (solid) for $7 < k \leq 12$.



(a) $k = 4, \bar{k} = 4$ (b) $k = 4, \bar{k} = 4$ (c) $k = 4, \bar{k} = 4$

(d) $k = 5, \bar{k} = 4$ (e) $k = 5, \bar{k} = 4$ (f) $k = 5, \bar{k} = 4$

(g) $k = 6, \bar{k} = 4$ (h) $k = 6, \bar{k} = 4$ (i) $k = 6, \bar{k} = 4$

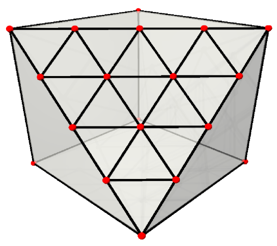
(j) $k = 7, \bar{k} = 4$ (k) $k = 7, \bar{k} = 4$ (l) $k = 7, \bar{k} = 4$

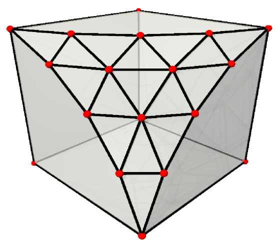
Figure 4.15: Aberration polytopes for solutions with maximal entries $(3.75, 3.75, 3.75, 3.75)$ (left), $(5.25, 4.25, 3.25, 2.25)$ (middle), and $(8, 4, 2, 1)$ (right) for $4 \leq k \leq 7$. For the balanced solution (left) we also depict the degenerated vertices.

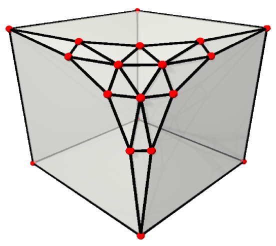
for any index $i > \bar{k}$. As we want to visualize the aberration polytope, we choose $m = 3$. Accordingly, k will range from 12 $(= m\bar{k})$ down to 4 $(= \bar{k})$, as below that threshold the effect of $\bar{k}$ vanishes.

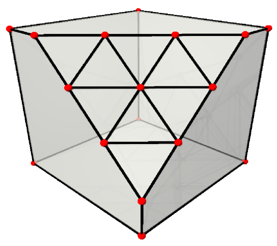
Figure 4.15[4] and Figure 4.16 show in the left column the aberration poly-

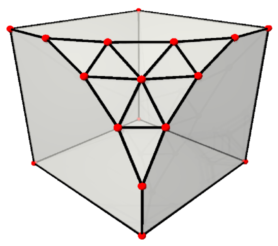
[4]For an animated comparison go to [55]. The images are created with the help of POLYMAKE [33].

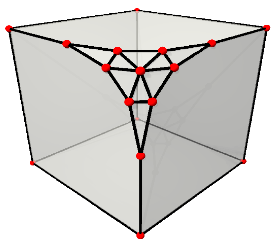

(a) $k = 8, \bar{k} = 4$

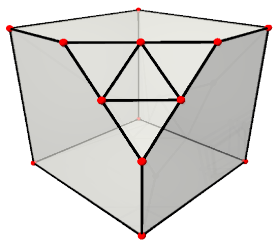

(b) $k = 8, \bar{k} = 4$

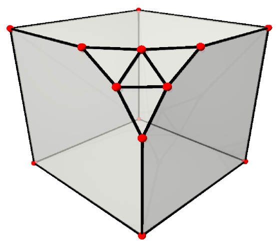

(c) $k = 8, \bar{k} = 4$

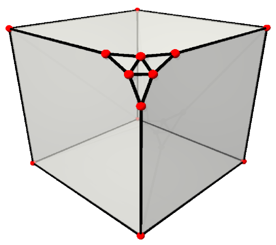

(d) $k = 9, \bar{k} = 4$

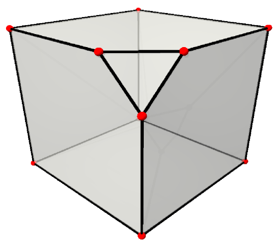

(e) $k = 9, \bar{k} = 4$

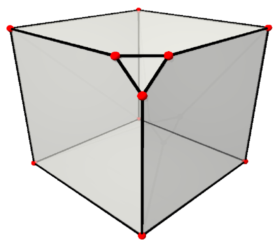

(f) $k = 9, \bar{k} = 4$

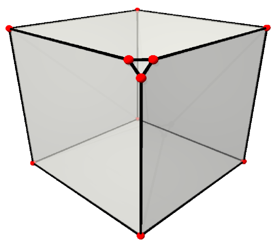

(g) $k = 10, \bar{k} = 4$

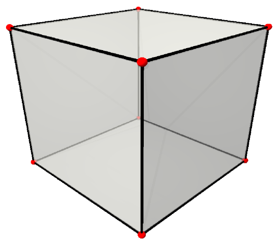

(h) $k = 10, \bar{k} = 4$

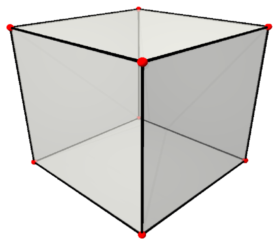

(i) $k = 10, \bar{k} = 4$


(j) $k = 11, \bar{k} = 4$


(k) $k = 11, \bar{k} = 4$


(l) $k = 11, \bar{k} = 4$


(m) $k = 12, \bar{k} = 4$


(n) $k = 12, \bar{k} = 4$

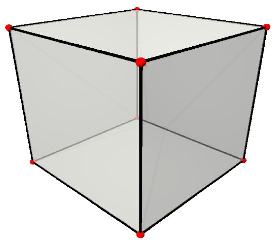

(o) $k = 12, \bar{k} = 4$

Figure 4.16: Aberration polytopes for solutions with maximal entries $(3.75, 3.75, 3.75, 3.75)$ (left), $(5.25, 4.25, 3.25, 2.25)$ (middle), and $(8, 4, 2, 1)$ (right) for $7 < k \leq 12$. For the balanced solution (left) we also depict the generated vertices.

topes of this planning solution for different k. The origin in the background of the pictures corresponds to the point $b - A^*x$ in the row space. A solution x is recovery robust if the whole polytope lies below the recovery hyperplane $\bar{R}$. The vertices of such a polytope correspond to the extremal choices of a_j as described in the above observation.

In the middle and right row of Figure 4.15 and Figure 4.16 we show the aberration polytope for the same values of k but two other planning solutions. In Figure 4.13 and Figure 4.14 the aberration polytopes of different planning solutions for the same k are drawn together. In each subfigure, the lowest polytope (with green vertices) corresponds to a vector x with maximal four entries equal to 3.75. Because all maximal four entries are equal, the combinatorics of these polytopes are simpler: Some of the vertices are degenerated. The middle polytope (with blue vertices) corresponds to a solution x with the following maximal entries: $\bar{x}_1 = 5.25, \bar{x}_2 = 4.25, \bar{x}_3 = 3.25$, and $\bar{x}_4 = 2.25$. Here the combinatorics are equal to those of the first solution, but the aberration polytopes contain those of the second planning solution and are contained in those of the first. Note that all solutions x have equal sum of the four greatest entries.

The planning solution influences heavily the shape of the aberration polytope. Two observations are apparent from the pictures:

- A more balanced solution causes less freedom for the adversary to deviate.

- The shape of the aberration depends on the relative values of the $\bar{k}$ maximal entries of x. Figuratively, the aberration can change from flat to ballooned, while the sum of the maximal values remains invariant. Therefore, an a priori, polyhedral analysis of the coincidental covering of scenarios as in the previous section seems impossible.

Strict Robustness Figure 4.13 and Figure 4.14 also illustrate an advantage of recoverable robustness in comparison to strict robustness. The strict robust approach cannot make use of the fact that there is a vertical limit k for the disturbance matrices $\tilde{A}$. In the perspective of strict robustness we always have $k = m\bar{k}$. The three solutions for which we draw the figures create significantly different aberration polytopes—except for $k = 12$, where the three polytopes coincide. But this is the only case that strict robustness can consider. The strict robust approach cannot distinguish between the robust qualities of the fully balanced solution (drawn solid) and the other solutions. Figuratively, a strict robust solver only sees three identical hypercubes as aberration polytopes of the three solutions. In contrast, in the perspective of the recovery robust solver the solutions cause three different, nested polytopes. This visualizes nicely the edge in information recoverable robustness has above the strict approach.

Tractability The important question is, whether one can handle this extra information efficiently. Is there a polynomial time algorithm to solve SLRPs with disturbed planning matrix limited in $S^{\bar{k},k}$? The answer is affirmative, and we will provide for it in the sequel.

As a first and direct approach we can enumerate the possible vertices of the aberration polytope. This approach will be polynomial in m (and in n and $\hat{n}$) but exponential in k (and $\bar{k}$). To establish this, we have to count the number of possible vertices of the aberration polytopes for given $k, \bar{k}$, and m. Remember that we characterized a set of candidates for the vertices independent of the planning solution x.

In a second step, we show that the linear program resulting from this enumeration can efficiently be separated. Thus, there is an algorithm for SLRPs over $S^{\bar{k},k}$ (k part of the input) with polynomial running time by the ellipsoid method. For practical purposes the given separation can be used for an approach by delayed constraint generation.

Lemma 4.31. *The aberration polytope of an SLRP with m-dimensional row space over $S^{\bar{k},k}$ has at most*

$$\phi(m,k) := 1 + \sum_{\ell=0}^{m} (-1)^{\ell} \binom{k+m-\ell r-1}{m-1} \binom{m}{\ell} \in \mathcal{O}(m^k)$$

vertices.

Proof. There are

$$\binom{k+m-1}{m} \in \mathcal{O}(m^k)$$

m-tuples of integers with sum equal to k. Now, we only have to throw out those tuples that have at least one integer greater than $\bar{k}$. For convenience set $\bar{k}+1 = r$. Consider the

$$\binom{k-r+m-1}{m}$$

m-tuples of integers with sum equal to $k-r$. Changing one of the m, say a_j, to $a_j + r$, guarantees that the resulting tuple will have at least one entry larger than $r-1 (= \bar{k})$. We have m choices for this change. So, we can subtract

$$m\binom{k-r+m-1}{m}$$

except that this way we subtract twice those tuples that have at least two entries greater than $r-1$. So, using an inclusion-exclusion argument we arrive at the formula

$$\sum_{\ell=0}^{m}(-1)^{\ell}\binom{k+m-\ell r-1}{m-1}\binom{m}{\ell}.$$

The last vertex is the origin. □

Separation In the scenario expansion of the SLRP we can now restrict the infinite scenario set $S^{\bar{k},k}$ to the finite set of $(\phi(m,k)-1)$-many potentially maximizing m-tuples of row sums $(a_i)_i$. Denote this set of m-tuples by $\Phi := \Phi(m,k)$. We can replace for each m-tuple $(a_i)_i$ the expression $\tilde{A}x$ by $(\sum_{\ell=1}^{a_i}\bar{x}_\ell)_i$. By the above lemma this yields a linear program of size in $\mathcal{O}\left(\phi(m,k)m(n+\hat{n})\right)$—called the *Aggregated Expansion of an SLRP over* $S^{\bar{k},k}$—with the same set of feasible solutions as the SLRP (and, of course, an identical objective).

$$\begin{aligned}
&min_{(x,\bar{x})\geq 0}c'x\\
&\qquad\qquad A^0x \geq b^0\\
&\qquad\qquad R(x,\bar{x}) \geq r\\
&\qquad\qquad \forall (a_i)_i \in \Phi :\\
&\qquad\qquad\qquad \left(A^*x - \left(\sum_{\ell=1}^{a_i}\bar{x}_\ell\right)_i\right)'\nu \geq H
\end{aligned}$$

with the constant H is the distance of the recovery plane from the original right-hand side b in direction of the recovery plane's outer normal vector ν. Further, the matrix R together with the vector r define a system of linear constraints which in a maximization framework yield a value of $\bar{x}_i$ equal to that of i-th largest entry of x.

For this linear program we can easily show the following lemma.

Lemma 4.32. *The Aggregated Expansion of an SLRP over $S^{\bar{k},k}$ can be separated in polynomial time.*

Proof. We show that, given the outer normal vector ν of the recovery plane and the maximal entries $\bar{x}$ of a solution x, a greedy algorithm can find the maximal vertex z_x^{max} of the aberration polytope with ν as objective function. If some scenario cannot be recovered for solution x, then its vector in the aberration polytope, and therefore also z_x^{max} must lie above the recovery plane. Hence, the algorithm either returns a scenario, i.e., a polynomial set of inequalities, containing a violated inequality, or proves that x is recovery robust.

We are given an $\bar{k}$-dimensional vector $\bar{x}$ and an m-dimensional vector ν, which is orthogonal to the recovery plane. We construct the $\bar{k} \cdot m$ many pairs $(\bar{x}_i, \nu_j)$ of their entries. To such a pair we assign a weight equal to $\bar{x}_i \cdot \nu_j$. To construct a maximal vertex, we choose k of these pairs with maximal sum of weights. A chosen pair $(\bar{x}_i, \nu_j)$ corresponds to the adversary changing maximally the coefficient of the i-th biggest entry in the j-th row of the planning matrix. As there are only $\bar{k}$ many pairs with the same ν_j, we have that in each row at most $\bar{k}$ entries can be changed. The resulting maximal set of pairs represents a potential vertex of the aberration polytope of x, which is either below and the closest to the recovery plane among all edges of the aberration polytope, or beyond the recovery plane.

Obviously, this problem is a matroid maximization, and can be solved by a greedy algorithm.

□

One can construct a different matroid problem, using Observation 4.30 and thereby avoid to generate all $\bar{k} \cdot m$ many pairs.

By the equivalence of separation and optimization [34] we get:

Theorem 4.33. *An SLRP over the scenario set $S^{\bar{k},k}$ can be solved in polynomial time.*

Back to Stochastic Programming We have seen that we can solve an SLRP with disturbed planning matrix. To this end we showed that programs of the following form can be separated efficiently:

$$\begin{aligned} \min\ & c'x \\ \text{s.t.}\ & Ax \geq b \\ & \Gamma_{(k,\bar{k},\hat{A},D)}x \geq \gamma_{(k,\bar{k},\hat{A},D)} \\ & x \geq 0 \end{aligned}$$

where $\Gamma_{(k,\bar{k},\hat{A},D)}$ and $\gamma_{(k,\bar{k},\hat{A},D)}$ define a set of linear inequalities which ensure a feasible solution x to be recoverable by the recovery matrix $\hat{A}$ with cost at most D in all scenarios of $S^{\bar{k},k}$.

Assume the management of a railway service provider sets up the following requirement. In case of few disturbances the total delay shall be very small, in case of several disturbances it may be a bit larger, and in case of enormously many disturbances in the system, the delay must still stay below a given, though large bound. This way the management can describe its risk aversion without reference to a distribution. We can incorporate such a requirement into our approach.

Let $\bar{k}$ and $\hat{A}$ be fixed and write $\Gamma_{(k,D)}$ and $\gamma_{(k,D)}$ for short. Assume we are given a multi-criteria planning objective in the form of pairs of a maximal total disturbance k and an acceptable corresponding recovery budget D_k for every $1 \leq k \leq k_{\max}$ and some large, upper bound $k_{\max}$. By the same separation algorithm as above we can solve the following program:

$$\begin{aligned} \min c'x & \\ \text{s.t.} \qquad & Ax \geq b \\ & \Gamma_{(1,D_1)}x \geq \gamma_{(1,D_1)} \\ & \vdots \\ & \Gamma_{(k_{\max},D_{k_{\max}})}x \geq \gamma_{(k_{\max},D_{k_{\max}})} \\ & x \geq 0 \end{aligned}$$

We have to call the separation oracle $k^{\max}$-many times separately for each robustness criterion, $\Gamma_{(i,D_i)}x \geq \gamma_{(i,D_i)}$. The resulting solution will for each level of disturbance fulfill a different guarantee to stay below a certain recovery cost.

To proceed one step further, think of the vector $(D_i)_i$ as a vector of variables instead of constants and introduce constant vectors $(\bar{D}_i)_i$ and $(C_i)_i$ with same dimension as $(D_i)_i$. Then

$$\begin{aligned} \min c'x + C'D & \\ \text{s.t.} \qquad & \\ & Ax \geq b \\ & \Gamma_{(k,D_k)}x \geq \gamma_{(k,D_k)}, \ \forall 1 \leq k \leq k_{\max} \\ & D_k \leq \bar{D}_k, \ \forall 1 \leq k \leq k_{\max} \\ & x \geq 0 \end{aligned} \tag{4.37}$$

allows to trade maximal recovery cost for different scenario sets. Note that Problem (4.37) is similar to a scenario aggregation of a 2-stage stochastic program: For a given probability measure μ set $C_k := \mu(S^k) - \mu(S^{k-1})$. Then the objective of Problem (4.37) resembles an expected value.

To make the comparison more explicit: Assume the complete scenario space S is finite. Create a singleton as scenario set $\{s^i\}$ for each element of S and set the corresponding cost C_i to the probability of that scenario. Then the objective $C'D$ is the expected value of the second stage cost of a 2-stage stochastic program.

Still, for non-singletons as scenario sets the interpretation of C as a vector of probabilities contains a subtle difficulty: By our above analysis we have seen that solutions will cover in general more scenarios than required within a certain recovery budget. This means, we will count recovery cost D_k for all scenarios with total disturbance equal to k, although some of them can be recovered at a lower cost, e.g., by D_{k-1}. This is the imprecision we were aiming to quantify in the polyhedral perspective. But we have seen that this analysis is not likely to succeed for the case of disturbed planning matrices.

As an advantage the above program still produces a solution that fulfills a *guarantee*, even several guarantees, in contrast to a stochastic objective function. Imposing $D \leq \bar{D}$ we guarantee for each k a certain maximal recovery cost D_k. This can be achieved without any knowledge about underlying distributions. The quality guarantees can be more valuable to a practitioner than a risk measure.

In this multi-criteria approach recoverable robustness can be used to combine the virtues of stochastic programming and robust optimization.

4.5.6 Overview of the results

The following diagram summarizes the central results in comparison to the classical models by Soyster and by Bertsimas and Sim.

	Soyster	Bertsimas-Sim	Recoverable Robustness
Tractability	LP of size $2n \times (m+n)$.	LP of size $(m+k+1) \times (m+k+n)$ with k the number of uncertain dates, i.e., $k \leq m \times n$.	Polynomial, quadratic size LP for right-hand side and S^1. Separable LP for matrix disturbances.
Right-Hand Side.	*	*	Suitable to exploit truncation of right-hand side scenario sets.
Integrated Protection	—	Horizontal.	Horizontal & Vertical.
Coincidental Covering	—	For a single row and non-hostile disturbances.	Same bounds as B-S for non-hostile disturbances. Further results for right-hand side hostile disturbances. For hostile matrix disturbances the solution can adapt to the scenario set.
Suitable Problems	All entries almost always slightly disturbed.	Few disturbed rows with many entries.	Right-hand side disturbances. Limited disturbances in many rows. Applications with recovery.

Tractability All models are solvable by polynomial size linear programs. The recoverable robust program with right-hand side disturbance is particularly compact for the scenario set S^1. This scenario set can also be used for practical situations in which a higher number of smaller disturbances is likely.

Right-hand Side Disturbances The two classical models are usually presented for matrix disturbances only. Still, right-hand side disturbances can be introduced to the model in a simple way. Replace for a program of the form $Ax \geq b$ each entry of b by the maximal value, which it may attain in any likely scenario. Of course, this is a conservative approach and for reasons of simplicity usually not mentioned. Recoverable robustness is the only non-trivial robust approach to right-hand side disturbances.

Integrated Protection Introducing recovery to robust models is an end in itself and yields an approach in between strict robustness and 2-stage stochastic programming. By this approach the concept of integrated protection in classical robustness can be enhanced significantly. Integrated protection means that the solution need not be protect against each data entry being *separately* disturbed, but against scenarios for which only a limited amount of *total* disturbance occurs. So far one could only construct an integrated protection against the disturbances within each row separately. Introducing recovery to robust programs opens the door for integrating the disturbances of all matrix entries. For right-hand side disturbances this vertical integration is the only fruitful integration. (There is only one right-hand side entry in each row, thus horizontal integration is vain.) Without vertical integration right-hand side models are over-conservative. Without recovery no vertical integration is possible.

Coincidental Covering Bertsimas and Sim were the first to give a priori lower bounds for the probability of robust solutions being feasible in larger scenario sets than those they are constructed for. This links robust optimization to chance constraints. Their bounds strongly rest on the non-hostility of the disturbances. These bounds are powerful only if all disturbances occur in a small, fixed set of rows.

We give a different type of analysis for this phenomenon of coincidental covering which holds for the whole system of inequalities and which even holds for hostile disturbances. In fact, this is the first analysis of coincidental covering for right-hand side disturbances.

The analysis shows that for matrix disturbances a recovery robust solution can adapt much more accurately to the given scenario set than a strict robust solution. Therefore, the recovery robust solution may (intentionally) avoid coincidental covering in order to enhance the objective value.

Suitable Problems Consider the following classical problem for linear programming. A plant can run n different production processes. The amount of time for which a certain process runs is expressed in the decision variable x_j. The inventory of the plant holds a certain quantity b_i ($1 \leq i \leq m$) for each of the m raw materials used in the processes. The amount of raw material i used in one time unit of process j is given by the matrix entry a_{ij} of the matrix A. In case a_{ij} is negative, the j-th process produces $|a_{ij}|$ units of material i in each time unit as. Further we are given some n-dimensional vector c, where c_i expresses the income we gain from running process i for one unit of time. In case we want to maximize our income, we should solve the problem $\max_{x \geq 0} c'x$ subject to $Ax \leq b$.

Next, we become aware that the data in A is slightly wrong. Each actual a^s_{ij} lies in a small interval around the given value a_{ij}. We can still solve the above optimization problem and wait for a natural adjustment: If some raw material is depleted before a process using it has complete its assigned time, then the process has to stop earlier. The resulting production may yield significantly less income than a production plan x constructed by the Soyster model. If almost no matrix entry will have the planned value, the Soyster model is a good and simple approach.

If only one the materials has changes in its corresponding matrix entries and the sum of these changes is bounded, then we can apply the Bertsimas-Sim model to the row of that material in order to get a better income than by the Soyster model.

If we can assume that each of the production processes will be planned for a time which is clearly larger than the adjustments in any scenario, then a negative recovery is always admissible. Assume further the recovery will cause extra costs (e.g., in manpower) if the total time that the processes run exceeds the work time that was planned. Impose an upper bound to the use of extra manpower. This problem can be formulated as an SLRP with disturbed planning matrix. The result gives a higher income than the previous models. Moreover, this model allows for vertical integration of the scenario set, which again can increase the income. Finally, the recoverable robust approach can also handle a different kind of uncertainty, which might be even more important than the one considered so far in this example: The actually stored amount of material i in the inventory may differ from the planning data b_i. This can also be modeled as an SLRP.

BIBLIOGRAPHY

[1] *Stochastic Programming*, Wiley, Chichester, 1994. 54

[2] *Introduction to Stochastic Programming*, Springer Verlag, New York, 1997. 54

[3] *Stochastic Programming, Handbooks in Operations Research and Management Science Volume 10*, North-Holland, 2003. 54

[4] *Special issue on robust optimization*, Mathematical Programming A **107**, no. 1-2 (2006). 53

[5] S. B. Aharon Ben-Tal and A. Nemirovski, *Extending scope of robust optimization: Comprehensive robust counterparts of uncertain problems*, Mathematical Programming A **107**, no. 1-2 (2006), pp. 63–89. 79

[6] K. Albers and F. Slomka, *An event stream driven approximation for the analysis of real-time systems*, in Proc. 16th Euromicro Conference on Real-Time Systems, 2004, pp. 187–195. 35

[7] K. Albers and F. Slomka, *Efficient feasibility analysis for real-time systems with edf scheduling*, in Proc. Conf. on Design, Automation and Test in Europe, 2005, pp. 492–497. 35

[8] S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty, *On Nash equilibria for a network creation game*, in Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA'06), 2006, pp. 89–98. 7, 8

[9] N. Alon, S. Hoory, and N. Linial, *The Moore bound for irregular graphs*, Graphs and Combinatorics **18**, no. 1 (2002), pp. 53–57. 24

[10] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden, *The price of stability for network design with fair cost alloccation*, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04), 2004, pp. 295–304. 8

[11] E. Anshelevich, A. Dasgupta, E. Tardos, and T. Wexler, *Near-optimal network design with selfish agents*, in Proceedings of 35th Annual ACM Symposium on Theory of Computing (STOC'03), 2003, pp. 511–520. 8

[12] T. Baker and M. Cirinei, *Brute-force determination of multiprocessor schedulability for sets of sporadic hard-deadline tasks*, in Proceedings of the 10th International Conference on Principles of Distributed Systems (PODC'07), 2007. 31

[13] T. P. Baker, *An analysis of EDF schedulability on a multiprocessor*, IEEE Trans. Parallel Distrib. Syst. **16**, no. 8 (2005), pp. 760–768. 36

[14] T. P. Baker and S. K. Baruah, *Schedulability analysis of multiprocessor sporadic task systems*, in Handbook of Real-Time and Embedded Systems, S. H. Son, I. Lee, and J. Y.-T. Leung, eds., CRC Press, 2007. 36

[15] V. Bala and S. Goyal, *A non-cooperative model of network formation*, Journal of Econometrics **68**, no. 5 (2000), pp. 1181–1229. 8

[16] A.-L. Barabási and R. Albert., *Emergence of scaling in random networks*, Science **286** (1999), pp. 509–512. 6

[17] S. Baruah and T. Baker, *Schedulability analysis of global edf.* Accepted for publication at Real-Time Systems Journal, 2008. 37

[18] S. K. Baruah, R. R. Howell, and L. E. Rosier, *Feasibility problems for recurring tasks on one processor*, Theor. Comput. Sci. **118**, no. 1 (1993), pp. 3–20. 35

[19] N. Baumann and S. Stiller, *The price of anarchy of a network creation game with exponential payoff*, in Proceedings of the 1st International Symposium on Algorithmic Game Theory (SAGT'08), 2008. 8

[20] A. Berger, U. Lorenz, R. Hoffmann, and S. Stiller, *TOPSU-RDM: A simulation platform for railway delay management*, in Proceedings of SimuTools, 2008. 67

[21] D. Bertsimas and M. Sim, *Robust discrete optimization and network flows*, Mathematical Programming B **98** (2003), pp. 49–71. 94

[22] D. Bertsimas and M. Sim, *The Price of Robustness*, Operations Research **52**, no. 1 (2004), pp. 35–53. 83, 84, 88, 90, 91, 93, 94, 95, 96, 128

[23] V. Bonifaci, A. Marchetti-Spaccamela, and S. Stiller, *A constant-approximate feasibility test for multiprocessor real-time scheduling*, in Proceedings of the 16th Annual European Symposium on Algorithms (ESA'08), 2008. 36

[24] G. Buttazzo, *Hard Real-time Computing Systems. Predictable Scheduling, Algorithms and Applications*, Springer, 2nd ed., 2004. 29

[25] A. Caprara, L. Galli, S. Stiller, and P. Toth, *Recoverable-robust platforming by network buffering*, Tech. Report ARRIVAL-TR-0157, ARRIVAL Project, 2008. 105

[26] S. Chakraborty, S. Künzli, and L. Thiele, *Approximate schedulability analysis*, in Proc. 23rd IEEE Real-Time Systems Symposium, 2002, pp. 159–168. 35

[27] J. Corbo and D. C. Parkes, *The price of selfish behavior in bilateral network formation*, in Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC'05), 2005. 7, 8

[28] E. D. Demaine, M. T. Hajiaghayi, H. Mahini, and M. Zadimoghaddam, *The price of anarchy in network creation games*, in Proceedings of the 26th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'07), 2007. 7

[29] M. L. Dertouzos, *Control robotics: The procedural control of physical processes*, in Proc. IFIP Congress, 1974, pp. 807–813. 35

[30] P. Erdös, D. Kleitman, and B. Rothschild, *Asymptotic enumeration of kn-free graphs*, in International Colloquium on Combinatorial Theory—Atti dei Convegni Lincei, 1976, pp. 19–27. 25

[31] A. Fabrikant, A. Luthra, E. M. C. H. Papadimitriou, and S. Shenker, *On a network creation game*, in Proceedings of the 22nd annual Symposium On Principles of Distributed Computing (PODC'03), 2003, pp. 347–351. 7, 8

[32] N. Fisher and S. K. Baruah, *A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines*, in Proc. 17th Euromicro Conference on Real-Time Systems, 2005, pp. 117–126. 35

[33] E. Gawrilow and M. Joswig, *polymake*. `http://www.math.tu-berlin.de/polymake/`. 133

[34] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, 1988. 138

[35] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, 1963 **58** (Journal of the American Statistical Association), pp. 13–30. 86

[36] M. O. Jackson, *A survey of models of network formation: Stability and efficiency*, in Group Formation in Economics: Networks, Clubs and Coalitions, G. Demange and M. Wooders, eds., Cambridge University Press, Cambridge, UK, 2004, ch. 1, pp. 11–57. 7

[37] M. O. Jackson and A. Wolinsky, *A strategic model of social and economic networks*, Journal of Economic Theory **71** (1996), pp. 44–74. reprinted in Networks and Groups: Models of Strategic Formation, edited by Dutta and Jackson, Springer–Verlag, Heidelberg 2003. 8, 11

[38] A. Kleywegt, A. Shapiro, and T. Homem-De-Mello, *The sample average approximation methode for stochastic discrete optimization*, SIAM Journal of Optimization **12**, no. 2 (2001), pp. 479–502. 97, 98

[39] E. Koutsoupias and C. H. Papadimitriou, *Worst-case equilibria*, in Proceedings of the 16th STACS, 1999, pp. 404–413. 7

[40] T. W. Lam and K.-K. To, *Trade-offs between speed and processor in hard-deadline scheduling*, in Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, pp. 623–632. 36

[41] J. Y.-T. Leung and M. L. Merrill, *A note on preemptive scheduling of periodic, real-time tasks*, Inf. Process. Lett. **11**, no. 3 (1980), pp. 115–118. 35

[42] C. Liebchen, *Periodic Timetable Optimization in Public Transport*, dissertation.de – Verlag im Internet, Berlin, 2006. 105

[43] C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller, *Recoverable robustness*, Tech. Report ARRIVAL-TR-0066, ARRIVAL-Project, 2007. 56

[44] C. Liebchen and S. Stiller, *Delay resistant timetabling*, Preprint 024–2006, TU Berlin, Institut für Mathematik, 2006. 62

[45] C. L. Liu and J. W. Layland, *Scheduling algorithms for multiprogramming in a hard-real-time environment*, J. ACM **20**, no. 1 (1973), pp. 46–61. 35

[46] M. A. Melendez-Jiminez, *Network formation and coordination: Bargaining the division of link costs.* Presented at 57th European Meeting of the Econometric Society, August 2002. `http://www.webdeptos.uma.es/THEconomica.net1.pdf`. 8

[47] J. F. Nash, *Non-cooperative games*, Annals of Mathematics **54**, no. 286–295 (2951). 7

[48] A. Nemirovski and A. Shapiro, *On complexity of stochastic programming problems*, tech. report, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2005. 97, 98

[49] A. Nemirovski and A. Shapiro, *On complexity of Shmoys-Swamy class of two-stage linear stochastic programming problems*, tech. report, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2006. 97

[50] C. A. Phillips, C. Stein, E. Torng, and J. Wein, *Optimal time-critical scheduling via resource augmentation*, Algorithmica **32**, no. 2 (2002), pp. 163–200. 30, 33, 36

[51] T. Schikinger, *Complete Subgraphs of Random Graphs*, PhD thesis, Technische Universität München , Fakultät für Informatik, 2003. 25

[52] P. Serafini and W. Ukovich, *A mathematical model for periodic scheduling problems*, SIAM Journal on Discrete Mathematics **2**, no. 4 (1989), pp. 550–581. 105

[53] D. B. Shmoys and C. Swamy, *Stochastic optimization is (almost) as easy as deterministic optimization*, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004), 2004. 97

[54] A. Soyster, *Convex programming with set-inclusive constraints and applications to inexact linear programming*, Operations Research **21**, no. 4 (1973), pp. 1154–1157. 53, 88

[55] S. Stiller, *Visualization of abberation polytopes.* `http://www.math.tu-berlin.de/coga/movie/abpoly/`. 133

[56] The Stochastic Programming Community, *Stochastic programming community home page.* `http://www.stoprog.org/`. 54

[57] M. Vromans, R. Dekker, and L. Kroon, *Reliability and heterogeneity of railway services*, European Journal of Operational Research **172** (2006), pp. 647–665. 62

[58] A. Watts, *A dynamic model of network formation*, Games and Economic Behavior **34** (2001), pp. 331–341. 8, 11