Discrete Optimization
Teaching and
Research Area

RWTH AACHEN UNIVERSITY

On the Hazmat Network Design Problem under Uncertainties

by Alexander Renneke

A Master Thesis in Mathematics
submitted to the Discrete Optimization Teaching and
Research Area

First Examiner: Prof. Dr. Arie M.C.A. Koster
Second Examiner: Prof. Dr. G. Walther
RWTH Aachen University
January 2025

Contents

Chapter 1

Introduction

One of the disadvantages of any highly developed nation is its dependence and production of a multitude of dangerous elements and compounds that pose a severe health risk to many people if handled improperly. Whether it is raw materials for the production of other goods or the waste of byproducts in factories, these hazardous materials (as defined in [12]) must be dealt with cautiously. One of the most critical phases of this handling is their transport between locations during which there are less means available to both avoid and react to any accidents. Most crucially, accidents can directly expose the population around the transport routes to the effects of these unsafe materials.
The protection of the populace from the transport risks of hazardous materials is the duty of the local and national governments, which employ regulations such as [7] to fullfill this task. One of their tools available for this is the ability to prohibit the transport companies from using certain streets that are deemed to risky for hazmat transport. This Hazmat-Transportation-Network-Design-Problem (HTP) that is at the center of this work deals with the question of which of the available streets exactly should be disabled to minimize the overall risk that arises from hazmat transportation.
The basic model used in this work was introduced in [17] and built on in various other works such as [9] which adds further limitations that greatly simplify the solution process, [1] which improves the model both in terms of realism and size, and [28] which similar to this work considers a robust version of the problem.
Other works of a similar nature include [11] which deals with the question of how exactly the transport risk should be quantified, [20] which considers various other tools the government can use for risk reduction, and [5] which considers alternate network limitations on the companies to steer hazmat transports towards less risky routes.
The remainder of the work is structured as follows: In chapter 2, general concepts of optimization are summarized to be used in the later chapters.

In chapter 3, a definition of the HTP is given and then extended to a robust version, the complexity of which will be analyzed in chapter 4. In chapter 5, the problem is then formulated as using Mixed Integer Program (MIP) and an approach for solving it is introduced. This approach will be implemented, as detailed in chapter 6, and analyzed in a computational study that makes up chapter 7. Finally, chapter 8 will give a conclusion and an outlook on avenues for further research.

Chapter 2

Theoretical Background

In this chapter, the terms and theoretical basics used in the later chapters will be introduced. Besides a short summary of path formulations, this will include an overview of the concepts used in robust optimization and bilevel optimization as needed for this work.

2.1 Paths and Path Formulations

In a directed Graph $G = (V, A)$ with vertices s and t, a (s,t)-path is given either as an sequence of vertices from s to t or as a sequence of the arcs between these vertices. A path is called simple if in the vertex-representation, no vertex occurs more than once. The problem of finding a shortest (s,t)-path consists of finding a path p that is minimal with regards to some arc cost function $c : A \to \mathbb{R}^+$ among all feasible (s,t)-paths, where $c(p) \coloneqq \sum_{a \in p} c(a)$. This problem can be modelled via the following Linear Program (LP) (see [27] p.48 ff.):

$$\min_{y} \quad \sum_{a \in A} c_a y_a \tag{2.1.1a}$$

$$s.t. \quad \sum_{a \in \delta^+(v)} y_a - \sum_{a \in \delta^-(v)} y_a = 0, \qquad \forall v \in V \backslash \{s,t\} \tag{2.1.1b}$$

$$\sum_{a \in \delta^+(v)} y_a - \sum_{a \in \delta^-(v)} y_a = 1, \qquad v = s \tag{2.1.1c}$$

$$\sum_{a \in \delta^+(v)} y_a - \sum_{a \in \delta^-(v)} y_a = -1, \qquad v = t \tag{2.1.1d}$$

$$y_a \geq 0 \qquad \forall a \in A \tag{2.1.1e}$$

where the variables are given by $\{y_a \in \mathbb{R}^+ | a \in A\}$ and where $y_a > 0$ exactly if arc a is part of the path. The terms $\delta^+(v)$ and $\delta^-(v)$ represent the outbound and inbound arcs of vertex v. The constraint (2.1.1b) equalizes

that exactly if an inbound arc is part of the graph, there is an outbound arc to continue the path. The special roles of the source and target node are adressed by contraints (2.1.1c) und (2.1.1d). Due to $c(a) \geq 0$ for all $a \in A$, the resulting path is simple. This problem is also referred to as Minimum-Cost-Flow-problem (with a flow value of 1). In case the arc (t, s) is part of the graph, this problem can be alternatively expressed by

$$\min_{y} \quad \sum_{a \in A} c_a y_a \; - c_{(t,s)} \tag{2.1.2a}$$

$$s.t. \quad \sum_{a \in \delta^+(v)} y_a - \sum_{a \in \delta^-(v)} y_a = 0, \qquad \forall v \in V \tag{2.1.2b}$$

$$y_{(t,s)} = 1, \tag{2.1.2c}$$

$$y_a \geq 0 \qquad \forall a \in A \tag{2.1.2d}$$

Note that the factor $c_{(t,s)}$ used in the objective function (2.1.2a) is actually a constant factor. Due to total unimodularity, the restriction $y_a \in \{0, 1\}$ is not necessary.

2.2 Optimization under Uncertainties

In classic optimization problems, the goal is to find an optimal solution for a given set of parameters, all of which are known exactly. In many real-world applications, however, some parameters can only be guessed or confined to a certain range of possible values. This could be because of a lack of information or to find robust solutions that remain optimal even under slight canges in circumstances. This is where optimization under uncertainties becomes relevant. A classical example is a shortest path problem where the exact cost of the arcs remains unknown.

The possible forms of uncertainty can be characterized by several aspects, most important is the question which parameters are uncertain (for the shortest path, this could either be the form of the graph itself, the source-target-vertices or the cost function). Following this is the question of what the set of possible values is for every uncertain parameter. Most commonly, these values are either in a given interval ($c_a \in [c_{\min}(a), c_{\max}(a)]$) or limited to a finite number of possibilities ($c_a \in \{c_1, \ldots, c_n\}$).

Additionally there is often an uncertainty budget Γ to limit the amount of uncertainty. In the case of uncertain arc lengths this could either limit the number of arc costs deviating from a standard value or limit the sum of all deviations.

Finally, uncertainties can differ in how the decision process of the uncertainty works, i.e. by which criteria a given solution is deemed good or optimal. Most often, this is either done by considering the worst-case scenario (also

referred to as robust optimization), meaning the uncertain parameters are set to one of the possible values in a way that hinders the objective the most, or a random distribution is applied such that the expectation value of the objective must be optimized (also called stochastic optimization). In the pessimistic version, it is not always trivial to find the set of "least optimal" values for a given solution in order to gauge its quality.
For more background on robust optimization, see e.g. [4].

2.3 Bilevel Optimization

Bilevel optimization deals with a class of problems consisting of two different subproblems with linked variables.
The first subproblem, often called upper level or leader problem, makes use of variables that form an optimal solution of the second subproblem, also referred to as lower level or follower problem. This lower level in turn is parameterized by the variables of the upper level.
A prominent example of bilevel problems is the toll setting problem, a two-player-game on a graph in which a leader chooses a set of arcs to tax and the follower subsequently chooses a minimum-cost path.
In this case, the optimal path of the follower is determined by the arc costs which depend on the leader decision, while the revenue the leader recieves hinges on which path the follower chooses. For more information on this problem, see e.g. [16].
One of the most efficient ways of solving bilevel problems is the reformulation as a single-level problem. However, this translation of constraints from follower to leader can cause the resulting single-level problem to be exponentially more complicated as the leader decision cannot be treated as fixed anymore in the context of finding the appropriate follower decision. Most crucially, formerly linear bilevel problems can become non-linear.

Example 1. *Consider the bilevel problem where the upper level is given by*

$$\begin{aligned} \max_{x \in \mathbb{R}} \quad & y \\ s.t. \quad & 0 \leq x \leq 1, \\ & y \in OPT(x) \end{aligned}$$

where $OPT(x)$ is the set of optimal solutions for the lower level

$$\begin{aligned} \min_{y \in \mathbb{R}} \quad & y \\ s.t. \quad & y = \sqrt{1 - x^2} \end{aligned}$$

While both levels are linear seperately (as $\sqrt{1-x^2}$ is treated as a constant in the lower level), the set of feasible solutions is part of the unit circle (see figure 2.1 and can not be expressed by a finite number of linear constraints.

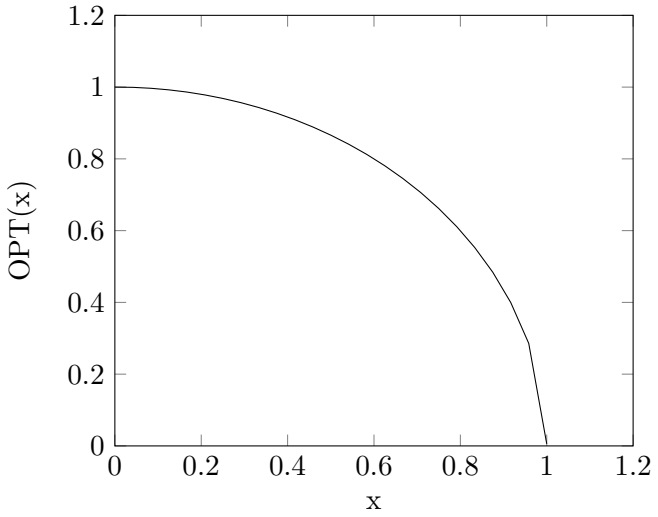

Figure 2.1: For every $0 \leq x \leq 1$ there is exactly one solution to the lower level $OPT(x) = \{\sqrt{1-x^2}\}$

In case a single-reformulation is not feasible, another solution approach is to utilize heuristics for either upper or lower level. However, in most cases no guranteed approximation factor can be given.
An important part of defining a bilevel problem is the level of cooperation between leader and follower: If the set of optimal follower decisions contains more than one solution, which one is chosen? In the optimistic case, it is the response most beneficial to the leader, whereas in the pessimistic case, it is the least beneficial one.

Example 2 (See [6])**.** *Consider the bilevel problem where the upper level is given by*

$$\begin{aligned} \min_{x \in \mathbb{R}} \quad & x + y \\ s.t. \quad & -1 \leq x \leq 1, \\ & y \in OPT(x) \end{aligned}$$

where $OPT(x)$ is the set of optimal solutions for the lower level

$$\begin{aligned} \min_{y \in \mathbb{R}} \quad & x \cdot y \\ s.t. \quad & 0 \leq y \leq 1 \end{aligned}$$

For $x < 0$, the optimal follower response is $y = 1$. For $x > 0$, it is $y = 0$. For $x = 0$, it is the interval $OPT(0) = [0, 1]$. (See figure 2.2)

Example 3 (Reformulation of example 2)**.** *Consider the single-level problem*

$$\begin{aligned} \min_{x,y\in\mathbb{R}} \quad & x+y \\ s.t. \quad & -1 \leq x \leq 1, \\ & x+y \geq 0, \\ & x+y \leq 1, \\ & y \in \{0,1\} \end{aligned}$$

In this reformulation, the tuple $(x,y) = (0,0)$ *is an optimal solution.*

$$\begin{aligned} \min_{y\in\mathbb{R}} \quad & x \cdot y \\ s.t. \quad & 0 \leq y \leq 1 \end{aligned}$$

For $x < 0$*, the optimal follower response is* $y = 1$*. For* $x > 0$*, it is* $y = 0$*. For* $x = 0$*, it is the interval* $OPT(0) = [0,1]$*. (See figure 2.2)*

Lastly, the combination of robust and bilevel optimization allows for the modelling of situations in which leader and follower do not have the same information. For example, the leader might be forced to work with uncertainties, whereas for a fixed leader decision the follower knows the exact values of these parameters.
For optimistic and pessimistic uncertainties, this can be modelled by introducing another "uncertainty"-player who acts after the leader has decided and before the follower decides. This new player can choose the fixed values of the uncertain parameters for the follower problem, either with the goal of helping or hindering the leader as much as possible.

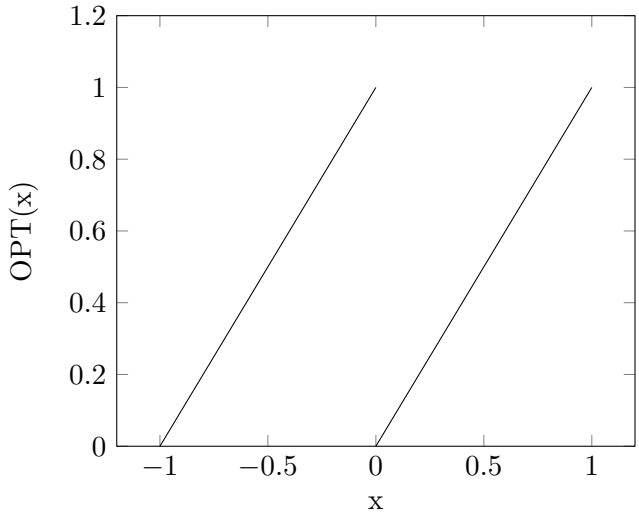

Figure 2.2: In case of $x = 0$, every $0 \leq y \leq 1$ is optimal for the follower. If it is not known which choice the follower will make, it can not be decided if $x = 0$ is an optimal solution for the leader.

Chapter 3

Problem Definition

In this chapter, a definition of the basic HTP is presented and compared to other variants used in related works. Afterwards, the given definition is expanded into a robust version that allows for uncertainty in the arc costs.

3.1 Basic Problem Definition

The problem definition largely follows the one presented in [17]. It is a bilevel problem dealing with a street network on which a set of hazadous materials must be transported from some source nodes to target nodes.
The follower in this case are the transport companies that choose the routes these transports take. Their priority during this is to minimize the time (or cost) needed for every transport. As those routes do not interfer with each other, it can be assumed without loss of generality that one central follower chooses every route simultanously.

Definition 1 (HTP Lower Level)**.** *Given a directed graph $G' = (V, A')$, an arc cost function $c : A \to \mathbb{R}_+$ and a set K of commodities with source nodes $\{s_k | k \in K\}$ and target nodes $\{t_k | k \in K\}$,*
find a set of paths $\{p_k | k \in K\}$ such that every path p_k is minimal among all (s_k, t_k)-paths in regards to

$$c(p) \quad := \quad \sum_{a \in p_k} c(a)$$

The leader of this bilevel problem is the government which is in charge of protecting the population from the dangers of accidents during these hazadous transports. One of the ways to do so is making sure these transport happen along routes that bear less risk (e.g. using less populated streets). While it can not dictate directly which paths the transporters should take, it has the authority to close off certain streets from being used for hazmat

transport.
For example, streets cutting through the middle of a city may be desireable to the follower when the alternatives are longer bypasses, but pose a high danger to the citizens when they used for hazmat transport. Therefore the government might want to prohibit their usage for such.
However, there are limits to the number and types of streets that can be banned simultanously such as disabling no more than half of all streets or big highways that would lead to unreasonably large delays when disabled.

Definition 2 (HTP Upper Level)**.** *Given a directed Graph $G = (V, A)$, the same parameters c and K as used in definition 1 as well as an arc risk function $r : A \to \mathbb{R}_+$, a subset $B \subseteq A$ of closeable streets and a leader budget $b \in \mathbb{N}$,*
find a subset $T \subseteq B$ with $|T| \leq b$ that minimizes $r(P)$, where $P = \{p_k | k \in K\}$ is one of the optimal solutions to the lower level of definition 1 parameterized by $G' = (V, A \backslash T)$ and where

$$r(P) \quad := \quad \sum_{p_k \in P} \sum_{a \in p_k} r(a)$$

Note that the risk of an arc is counted multiple times if more than one transport is conducted along the represented street. Note also that this definition presumes cooperation between leader and follower, i.e. if multiple optimal solutions for the lower level exist, the leader can assume the one resulting in the least risk will be chosen by the follower.

3.2 An Example of the Basic Problem

For example, consider the instance given in 3.1 for a leader budget of $b = 3$. If no arcs are removed, the follower will choose the paths along the central arc for both commodities as they have a cost of 3 versus a cost of 7 and 12 for the upper and lower path respectively. This leads to a risk of $3+12 = 15$. On one hand, removing only the upper or lower arc leads to the same follower decision and therefore the same risk. On the other hand, removing the middle arc together with the upper or lower arc is not feasible, since this makes either t_1 or t_2 unreachable.
Therefore the only impactful change is removing the middle arc while keeping the other arcs. In this case, the paths along the upper and lower arc must be chosen, which leads to a combined risk of $7 + 3 = 10$. This decision is therefore optimal.
Note that for this choice, the risk of p_1 is actually higher. This shows it is not sufficient to find minimum-risk paths for every commodity individually,

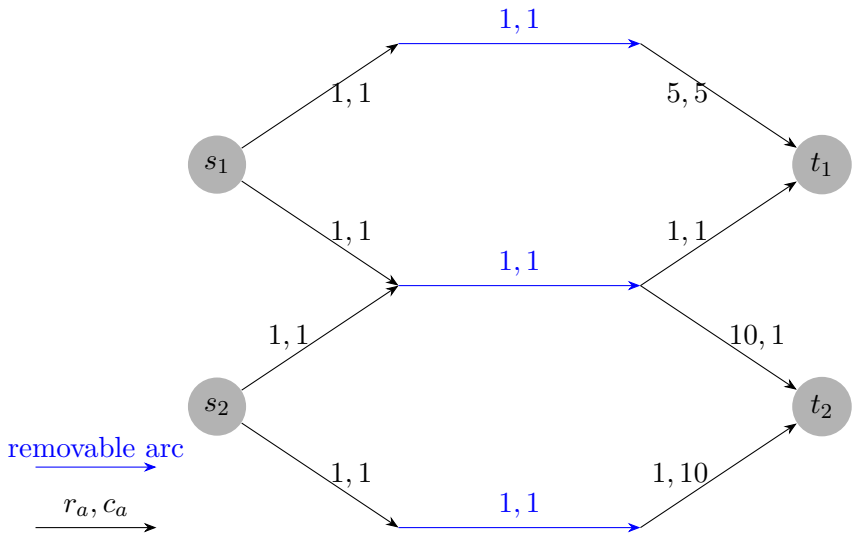

Figure 3.1: Example of a graph with 2 commodities. Removal of the middle arc increases the risk of the (s_1, t_1)-path by 4 but decreases the risk of the (s_2, t_2)-path by 9 and is therefore optimal.

as the leader can not restrict the remaining arcs to be used by a specific commodity only. While the follower can consider each commodity seperately, the leader can not do the same.

The aforementioned fact that the removal of the upper or lower arc while keeping the middle arc has no effect can be generalized by the following lemma:

Lemma 1 (Lemma)**.** *Let $G = (V, A)$ be a graph for which the follower chooses the cost-minimal paths $P = \{p_k | k \in K\}$. Then, if $T \cap p_k = \emptyset$ for all $p_k \in P$ and T is chosen by the leader in problem 2, the paths P are also chosen by the follower in problem 1 with $G' = (V, A \backslash T)$.*

Proof. Obiously, the removal of arcs has no effect on the cost of the paths in G. Since it also does not enable any new paths, the previously chosen paths remain cost-minimal among those in G'. Due to $T \cap p_k = \emptyset$ for all $k \in K$, the set P continues to be a valid choice of paths in G'. □

In other words, if the follower response on the unrestricted path is known, the resulting overall risk is only changed (either positively or negatively) if the leader removes at least one of the arcs normally chosen by the follower. The reverse however is not true, as it is not sufficient to limit T to the minimum-cost paths of the unrestricted case.

Another consideration for the given example is that, if the arc with a cost of 10 had a cost of 1 instead, the optimal solution would depend on the cooperation between leader and follower. As the (s_2, t_2)-paths along the lower and the middle arc would have the same cost of 3 but differing risk (3 vs 12), the follower can act either leader-friendly or antagonistic. In the former case, the optimal leader decision would be to not remove any arcs, while in the latter case, the middle arc should be eliminated.

3.3 Comparison to Other Models

While the overall structure of the previous definition is the same as [17], it differs in two aspects. It is less detailed in that its arc risk values are not commodity-specific, which would allow to denote more or less dangerous materials or streets where some materials can lead to more dangerous accidents than others. It is more detailed however in terms of limiting the leader decision to avoid-overregulation. In the cited work, the closure of a street has no effect on the leaders ability to close further streets.
In contrast, [1] and [9] differ from the given definition by introducing an additional demand value for every commodity by which the risk and cost of the corresponding path is scaled. This can be replicated in the given definition by introducing multiple commodities that share both source and target node altough it results in a significantly larger instance. The latter of the two cited works also lacks the limitations imposed on the arc choice of the leader.
Another work, [26], introduces several modes of transportation such as trains and trucks. In their model, the leader can close off the streets for specific transportation modes, while the follower can also choose by which mode(s) the commodites should be moved in addition to the choice of path. Again, in this works problem definition the leader is able to close an unlimited number of arcs simultanously.

3.4 Arc Cost Uncertainty

The arc cost in the problem definition is used to represent the time needed to traverse the corresponding street. Unlike the strategic (long-term) decision of the government which streets are generally closed for hazmat transport, the transport companies decision of which path to take need only be made directly before the actual transport. As such, it can be influenced by day-to-day factors affecting the travel time, such as increased traffic or bad weather. Consequently, the actual time needed for traveling along the street, i.e. the arc cost, used by the follower in his route calculations is not exactly known to the leader. While the leader might be able to find out the time needed under optimal conditions and the time needed under less-than-ideal condi-

tions, he can not know in advance which of the two cases applies to any of the streets at the time the transport routes are picked. Thus, it is expedient to expand the given problem definition to include arc cost uncertainty.
When considering worst-case uncertainty, i.e. the objective to minimize the overall risk even under the worst circumstances, it is important to know how this worst-case scenario can be found. This search can also be defined as a bilevel problem where the upper level deals with the realization of the uncertainty, i.e. deciding which of the arcs should have standard cost and which ones should have increased cost. Based on this decision the lower level consists of choosing the best (cost-minimal) paths in adherance to the modified arc costs.

Definition 3 (Cost Uncertainty Lower Level)**.** *Given a directed graph* $G' = (V, A')$*, an arc cost function* $c : A \to \mathbb{R}_+$*, a cost increase function* $c^+ : A \to \mathbb{R}_+$*, a subset* T' *of arcs with increased cost and a set* K *of commodities with source nodes* $\{s_k | k \in K\}$ *and target nodes* $\{t_k | k \in K\}$*,*
find a set of (s_k, t_k)*-paths* $\{p_k | k \in K\}$ *that are minimal in regards to the adjusted cost*

$$c'(p_k) \quad := \quad \sum_{a \in (p_k \setminus T')} c(a) \; + \; \sum_{a \in (p_k \cap T')} c(a) + c^+(a)$$

Note that in practice this definition is equal to the basic case albeit with a more convoluted notation of the arc costs. The problem of finding the worst case scenario for arc cost uncertainty consists of finding the arc subset T' used in the previous definition.

Definition 4 (Cost Uncertainty-Subproblem (CUS))**.** *Given the same parameters* G', c, c^+ *and* K *as used in definition 3, as well as an arc risk function* $r : A \to \mathbb{R}_+$ *and an uncertainty budget* $\Gamma \in \mathbb{N}$*,*
find a subset $T' \subseteq A$ *with* $|T'| \leq \Gamma$ *that maximizes* $r(P)$*, where* $P = \{p_k | k \in K\}$ *is an optimal solution to the* T'*-parameterized lower level of definition 3 and*

$$r(P) \quad := \quad \sum_{p_k \in P} \sum_{a \in p_k} r(a)$$

Note that this definition also assumes cooperation between uncertainty player and follower. In this context it means that among multiple sets of cost-minimal paths, the one resulting in the highest risk will be selected by the follower.
In turn, the task of the government is to minimize this worst-case risk. This can be defined as a bilevel problem, where the lower level is a bilevel problem itself, resulting in three levels overall.

Definition 5 (HTP Cost Uncertainty Upper Level)**.** *Given the same parameters G, B and b as used in definition 2 and c, c^+, K, r and Γ as used in definition 4,*
find a subset $T \subseteq B$ with $|T| \leq b$ that minimizes r^*, where r^* is the optimal solution value to the uncertainty level of definition 4 parameterized by $G' = (V, A \backslash T)$.

Unlike the basic definition, it is not necessary to consider cooperation between the leader and the uncertainty player as their goals are diametrically opposed. Part of the worst case-mentality is the assumption that the follower also acts antagonistic, i.e. acts in the best interest of the uncertainty player.

3.5 Uncertainty in Related Works

The definition of cost uncertainty can be transferred to risk uncertainty quite easily. A noteable difference is that the follower is no longer affected by the realization of uncertainty, meaning the question of which arcs should have increased risk can be answered after finding an optimal follower response to a given leader decision.
An example for another work focussing on arc risk uncertainty is [22], which is also based on the basic problem given in [17]. One of the biggest differences is that the type of uncertainty used in the work is based on intervalls (i.e. $r_a \in [r_{\min}, r_{\max}]$ instead of $r_a \in \{r_{\min}, r_{\max}\}$). Additionally, a consideration towards extensions accounting for commodity-specific risk uncertainty is made.
Similarly, the work [28] considers the HTP with uncertain arc risks. It deviates from the work mentioned before in that it does not restict the total uncertainty by a uncertainty budget Γ.

Chapter 4

Complexity of the Cost Uncertainty Subproblem

In the following chapter, it is shown that the CUS per definition 4 is NP-hard even when limited to a single commodity. This is accomplished by showing that it is part of NP and that any instance of the longest path decision problem can be solved in polynomial time if the same holds true for the CUS.

Definition 6 (Longest Path Decision Problem)**.** *Given a graph $G' = (V, A')$, an arc distance function $d : A' \to \mathbb{R}^+$, two vertices s and t and some threshold $m \in \mathbb{N}$,*
decide if there is an (s,t)-path p such that

$$\sum_{a \in p} d(a) \ \geq \ m$$

Problem 6 is NP-hard as shown in ([13], p. 213). Since this problem is defined as a decision problem, a decision version of the CUS is needed for comparison.

Definition 7 (CUS Decision Problem)**.** *Given a graph $G' = (V, A')$, an arc risk function $r : A' \to \mathbb{R}^+$, an arc cost function $c : A' \to R^+$, two vertices s and t and some threshold $m \in \mathbb{N}$,*
decide if there is an arc subset T' and an (s,t)-path p which is minimal regarding

$$c'(p) \quad \coloneqq \quad \sum_{a \in p} c(a) \ + \sum_{a \in (p \cap T')} c^+(a)$$

and which fullfills

$$\sum_{a \in p} r(a) \ \geq \ m$$

The first part to be shown is that a non-deterministic Turing-machine can solve problem 7 in polynomial time. Even though the given problem definition does not include a commodity set K or an uncertainty budget Γ, the following proof can be easily adapted to account for their addition.

Lemma 2. *Problem 7 is part of the class of $\mathcal{NP}$-problems.*

Proof. In the case of non-deterministic machines, the correct subset T' (with a size limited by $|A'|$) and the correct path p (also limited by $|A'|$) can be "guessed". Afterwards, the validity of T' and subsequently, the validity of the path, its cost-minimality and its resulting risk (and comparison with m) can be calculated in polynomial time each. □

For the other part of the proof of NP-hardness, it must be shown that any instance of problem 6 can be transformed into an equivalent instance of problem 7 of similar size in polynomial time.

Lemma 3. *If instances of problem 7 can be solved in polynomial time, the same holds true for instances of problem 6.*

Proof. Let (G, d, s, t, m) be an instance of problem 6. Construct an instance $(G', r, c, c^+, s', t', m')$ of problem 7 in the following way:

$$\begin{aligned}
G' &:= G, \\
r(a) &:= d(a) \qquad \forall a \in A, \\
c(a) &:= 0 \qquad \forall a \in A, \\
c^+(a) &:= 1 \qquad \forall a \in A, \\
s' &:= s, \\
t' &:= t, \\
m' &:= m
\end{aligned}$$

If this is a yes-instance and there is an arc subset T' and a path p with $r(p) \geq m$, the same path can be used for the original instance, since $d(p) = r(p) \geq m$ due to $d(a) = r(a)$ for all arcs $a \in A'$.
Equally, if there is a path p such that $d(p) \geq m$ in the original instance, choosing the same path in G' and $T' := A' \setminus p$ leads to p being the only (s,t)-path with $c'(p) = 0$ while also fullfilling $r(p) \geq m$.
Thus, both instances are equivalent and, since the transformation is polynomial in both time and size, if the transformed instance can be solved polynomially so can the original one. □

See figures 4.1 and 4.2 for an example of the instance transformation given above. Combining these two theorems leads to the desired result of this chapter.

Theorem 1. *The CUS is NP-hard.*

Proof. Follows directly from lemma 2 and lemma 3. □

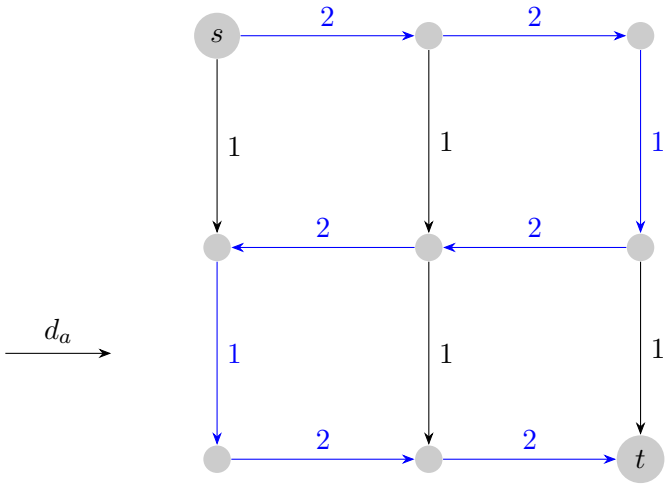

Figure 4.1: Instance of the longest path problem with fixed source and target node. The longest path with a distance of 14 is along the blue arcs.

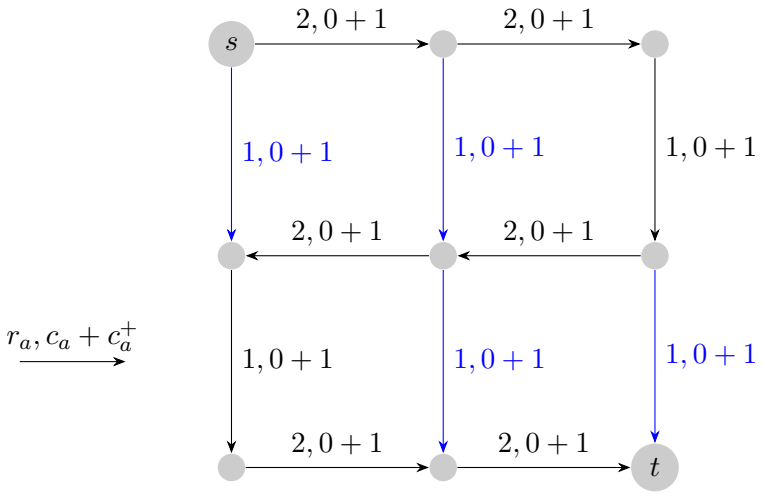

Figure 4.2: Instance of the cost uncertainty subproblem. The maximal risk of 14 can be archieved by increasing the cost of the blue arcs.

Chapter 5

Formulation

In this chaper, the basic HTP as defined in the previous chapter is formulated as a combination of MIPs. After a brief overview of solution strategies for the basic case, the formulation is expanded to account for arc cost uncertainty. For this expanded formulation, a solution strategy is then introduced which will be implemented and tested in the subsequent chapters.

5.1 Formulation of the Basic Problem

Since the definition for the basic HTP is given as a bilevel problem, it is expedient to use a formulation that models every level as an MIP and then link the two Programs. As the lower level according to definition 1 is at its core a shortest path-problem, the formulation given in chapter 2 can be utilized. Thus the lower level can be written as

$$\min_{y} \quad \sum_{k\in K}\sum_{a\in A} c_a y_{a,k} \tag{5.1.1a}$$

$$s.t. \quad \sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = 0, \qquad \forall k \in K,\ \forall v \in V\backslash\{s_k, t_k\} \tag{5.1.1b}$$

$$\sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = 1, \qquad \forall k \in K,\ v = s_k \tag{5.1.1c}$$

$$\sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = -1, \qquad \forall k \in K,\ v = t_k \tag{5.1.1d}$$

$$y_{a,k} \leq x_a \qquad \forall k \in K,\ \forall a \in A \tag{5.1.1e}$$

$$y_{a,k} \in \{0,1\} \qquad \forall k \in K,\ \forall a \in A \tag{5.1.1f}$$

Here, the variable $y_{a,k}$ is 1 if arc a is used as part of the transport route for commodity k and 0 if not. Due to total unimodularity, the constraint (5.1.1f) can be replaced by $y_{a,k} \geq 0$. The constant c_a used in the objective function (5.1.1a) is the cost associated with arc a.
The term x_a used in constraint (5.1.1e) is also treated as a constant, even though in the upper level this is a binary variable with $x_a = 0$ if the leader prohibits use of this arc and 1 if not.
Note that in the case of $|K| = 1$, an equivalent formulation would be

$$\min_{y} \quad \sum_{k \in K} \sum_{a \in A} c_a y_{a,k} \; - \sum_{k \in K} c_{t_k,s_k} \tag{5.1.2a}$$

$$s.t. \quad \sum_{a \in \delta^+(v)} y_{a,k} - \sum_{a \in \delta^-(v)} y_{a,k} = 0, \qquad \forall k \in K,\ \forall v \in V \tag{5.1.2b}$$

$$y_{t_k,s_k} = 1 \qquad \forall k \in K \tag{5.1.2c}$$

$$y_{a,k} \leq x_a \qquad \forall k \in K,\ \forall a \in A \tag{5.1.2d}$$

$$y_{a,k} \geq 0 \qquad \forall k \in K,\ \forall a \in A \tag{5.1.2e}$$

For a single commidity, the arc (t_k, s_k) can be added if missing as it is never part of a minimum-cost path. For multiple commodities however, adding the arc can enable new paths for the other commodities. Similarily, the constraints (5.1.2c) and (5.1.2d) prevent the leader from removing the arc if it is a part of the graph already.
Assuming the leader decision is fixed, this can be adressed in the following way: If the arc exists and is usable, nothing needs to be done. If it either does not exist or is disabled by the leader, it can be added/enabled with a cost of N that is high enough to ensure no other commodity is transported along this arc.

The upper level as defined in 2 can be formulated via the Integer Program

$$\min_{x,y} \quad \sum_{\substack{k \in K, \\ a \in A}} r_a y_{a,k} \tag{5.1.3a}$$

$$s.t. \sum_{a \in A} x_a \geq |A| - b \tag{5.1.3b}$$

$$x_a = 1 \qquad \forall a \in A \backslash B \tag{5.1.3c}$$

$$x_a \in \{0,1\} \qquad \forall a \in A \tag{5.1.3d}$$

$$y \in \mathcal{OPT}(x) \tag{5.1.3e}$$

As mentioned above, the variables x_a represent the leader's decision on which arcs should be usable for hazmat transport. The constants r_a in the objective function (5.1.3a) denotes the arc risks. Constraint (5.1.3b) limits the number of disabled arcs to b, whereas B in constraint (5.1.3c) represents the set of all removable arcs. The term $\mathcal{OPT}(x)$ in constraint represents the set of optimal solutions to the lower level given by formulation (5.1.1) which is dependent on x.

5.2 Solving the Basic Problem

Finding optimal solutions to the basic HTP as formulated above is the focus of several works by other authors, therefore instead of a new approach to the basic problem, a few examples of existing methods will be presented.
The work which introduces the bilevel-definition of the HTP, [17], takes advantage of the fact that the lower level is unimodular and can be therefore treated as a Linear Program. Due to this, the authors are able to leverage the Karush-Kuhn-Tucker (KKT) conditions of the lower level to transform it from an optimization problem to a feasiblity problem. By eliminating the lower level objective via additional constraints, the bilevel problem can be reformulated into a single-level problem.
The problem of this process is the loss of linearity which requires further adjustments utilizing "a large number" R. The resulting program can then be processed by commercial solvers for MIP problems.
This approach is further refined in [9]. In addition to the single-level reformulation, the authors impose additional demands to decrease the size of their formulation. By stipulating the set of available arcs has the form of a (Steiner) tree, they ensure there exists exactly one feasible path for every commodity. To reduce the transport cost of the follower, the authors then supplement the resulting tree with additional arcs that are chosen heuristically.
A subsequent work, [10], concentrates solely on heuristics to solve the bilevel problem. In this approach, an initial solution is given by the union of paths that are risk-minimal for all commodities individually. As seen in the example given in chapter 3, this is not necessarily an optimal solution. Based on this initial solution, the authors then identify arcs whose removal results in the best risk decrease.
A different method of solving the basic HTP is used in [25]. It uses a single-level formulation in which variables based on P^k are used, where P^k is a list of all possible routes for commodity k, sorted by cost in ascending order. The main obstacle of this is the exponential size of P^k for every $k \in K$. By including a sort of compromise between leader and follower such that the worst paths in the original graph (e.g. "more than twice the cost of the shortest path" or "more cost than the Kth shortest path") are not consid-

ered feasible, this problem can be mitigated.
Finally, the authors of [1] show another way of reformulating the bilevel-problem as a single-level problem. Instead of the KKT conditions, they make use of the dual of the linear follower problem to replace its objective function, which compared to [17] reduces the number of necessary binary variables. Another notable feature is that their formulation of the lower level objective function makes sure that the follower prefers paths of higher risk among paths of equal cost.

5.3 Formulation of Arc Uncertainty

Similar to the formulation of the leader and follower level as MIPs, the additional arc uncertainty level as detailed in definition 4 can be expressed via another MIP as

$$\max_{\gamma,y} \quad \sum_{k\in K}\sum_{a\in A} r_a y_{a,k} \tag{5.3.1a}$$

$$s.t. \quad \sum_{a\in A} \gamma_a \leq \Gamma \tag{5.3.1b}$$

$$\gamma_a \in \{0,1\} \qquad \forall a \in A \tag{5.3.1c}$$

$$y \in \mathcal{OPT}'(x,\gamma) \tag{5.3.1d}$$

As before, the variable $y_{a,k}$ equals 1 if arc a is part of the path for commodity k and 0 otherwise. The objective 5.3.1a serves to maximize the overall risk. The binary variable γ_a indicates whether the cost of arc a should be increased or not. Per constraint (5.3.1b), the number of arcs with increased cost is limited by the uncertainty budget Γ. The term $\mathcal{OPT}'(x,\gamma)$ denotes the set of optimal solutions for the follower level of the arc uncertainty problem, parameterized by x and γ. Note that x, which is a variable in the upper level, is treated as a constant at this level.
The single difference in the follower level between basic and robust problem is the addition of γ when calculating the arc costs. The modified objective function to replace objective (5.1.1a) can be written as

$$\min_{y} \quad \sum_{k\in K}\sum_{a\in A}(c_a \; + \; c_a^+\gamma_a)y_{a,k} \tag{5.3.2a}$$

while the constraints (5.1.1b)-(5.1.1f) remain unchanged. Analogously, from the perspective of the follower γ is treated as a constant and the follower level remains linear. In a similar vein, the leader level of the original problem requires the adjustment of constraint (5.1.3e) such that y (and γ) are an optimal solution of the uncertainty level instead of the basic follower level, meaning

$$(\gamma, y) \in \mathcal{OPT}'(x) \tag{5.3.3e}$$

In summary, the HTP Cost Uncertainty Problem per definition 5 can be written as

$$\min_{x} \quad \sum_{\substack{k\in K,\\ a\in A}} r_a y_{a,k} \tag{5.3.4a}$$

$$s.t. \sum_{a\in A} x_a \geq |A| - b \tag{5.3.4b}$$

$$x_a = 1 \qquad \forall a \in A\backslash B \tag{5.3.4c}$$

$$x_a \in \{0,1\} \qquad \forall a \in A \tag{5.3.4d}$$

$$(\gamma, y) \in \mathcal{OPT}'(x) \tag{5.3.4e}$$

where $\mathcal{OPT}'(x)$ refers to the optimal solutions to the problem defined in formulation (5.3.1) and the term $\mathcal{OPT}'(x, \gamma)$ refers to the optimal solutions of

$$\min_{y} \quad \sum_{k\in K}\sum_{a\in A}(c_a \ + \ c_a^+\gamma_a)y_{a,k} \tag{5.3.5a}$$

$$s.t. \quad \sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = 0, \qquad \forall k \in K, \ \forall v \in V\backslash\{s_k, t_k\} \tag{5.3.5b}$$

$$\sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = 1, \qquad \forall k \in K, \ v = s_k \tag{5.3.5c}$$

$$\sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = -1, \qquad \forall k \in K, \ v = t_k \tag{5.3.5d}$$

$$y_{a,k} \leq x_a \qquad \forall k \in K, \ \forall a \in A \tag{5.3.5e}$$

$$y_{a,k} \geq 0 \qquad \forall k \in K, \ \forall a \in A \tag{5.3.5f}$$

Overall, this formulation includes $\mathcal{O}(|A||K|)$ variables and $\mathcal{O}(|K|(|V| + |A|))$ constraints spread across three linked MIPs.

5.4 Solving the Uncertainty Problem

Similar to the basic case, the question arises how this formulation of the robust HTP can be solved efficiently. In this work, a two-fold approach is

presented. On one hand, a reformulation of the uncertainty and follower levels into a single-level MIP is given. On the other hand, heuristics for the leader level are introduced. An implementation of this approach is detailed and tested in the subsequent chapters.
The idea for the follower level is inspired by [25] and is based on the fact that finding a minimum-cost path is equivalent to finding a path p such that no other path has a lower cost than p. Since the actual value of an optimal solution is not needed, we can utilize the following linear program

$$\min_{y} \quad 1 \tag{5.4.1a}$$

$$s.t. \quad \sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = 0, \qquad \forall k \in K,\ \forall v \in V\backslash\{s_k, t_k\} \tag{5.4.1b}$$

$$\sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = 1, \qquad \forall k \in K,\ v = s_k \tag{5.4.1c}$$

$$\sum_{a\in\delta^+(v)} y_{a,k} - \sum_{a\in\delta^-(v)} y_{a,k} = -1, \qquad \forall k \in K,\ v = t_k \tag{5.4.1d}$$

$$y_{a,k} \leq x_a \qquad \forall k \in K,\ \forall a \in A \tag{5.4.1e}$$

$$y_{a,k} \geq 0 \qquad \forall k \in K,\ \forall a \in A \tag{5.4.1f}$$

$$\sum_{a\in A}(c_a + c^+_{a\gamma_a})y_{a,k} \ \leq \ \sum_{a\in p}(c_a + c^+_a\gamma_a) \qquad \forall k \in K,\ \forall p \in P^k(x) \tag{5.4.1g}$$

where $P^k(x)$ denotes the set of all feasible paths for commodity k that do not use arcs disabled by x. Apart from (5.4.1a) and (5.4.1g), this matches formulation (5.3.5). Due to the constant objective (5.4.1a), every feasible solution is also optimal. The similarity to the lower level results in the following property:

Lemma 4. *The set of optimal solutions for formulation* (5.4.1) *is the same as the set of optimal solutions for the formulation* (5.3.5).

Proof. Let y be a feasible solution for 5.3.5.
If y is not feasible (and therefore optimal) for formulation 5.4.1, there exists a commodity k' and a path $p' \in P^{k'}(x)$ such that

$$\sum_{a\in A}(c_a + c^+_a\gamma_a)y_{a,k'} \ > \ \sum_{a\in p'}(c_a + c^+_a\gamma_a)$$

Define an alternative solution y' via

$$y'_{a,k} := \begin{cases} 1, & \text{if } \text{a} \in p',\ k = k' \\ 0, & \text{if } \text{a} \notin p',\ k = k' \\ y_{a,k}, & \text{else.} \end{cases}$$

Then y' is a feasible solution for formulation 5.3.5. At the same time, its total cost is less than y due to the inequality above. Therefore, y can not be optimal for formulation 5.3.5.
Conversely, if y is not an optimal solution for 5.3.5, there exists a feasible solution y' with lower cost. Let $\{p_k | k \in K\}$ be the paths induced by y' (as noted in chapter 2). Then there exists at least one commodity $k' \in K$ such that

$$\sum_{a \in A} (c_a + c_a^+ \gamma_a) y_{a,k'} \ > \ \sum_{a \in A} (c_a + c_a^+ \gamma_a) y'_{a,k'} \ = \ \sum_{a \in p_{k'}} (c_a + c_a^+ \gamma_a)$$

and therefore y violates the constraint corresponding to $p_{k'}$ in 5.4.1g and can not be optimal for formulation 5.4.1. □

The advantage of this path-based formulation is the fact that the constraint (5.3.1d) used in the uncertainty level can be replaced by the constraints (5.4.1a)-(5.4.1g) to create a single-level formulation. The main disadvantage however is the size of the additional constraint (5.4.1g). Another disadvantage is the loss of linearity in (5.4.1g) as γ is not a constant on the uncertainty level. The latter problem can be circumvented by the introduction of auxiliary variables $\{\pi_{a,k} | a \in A, k \in K\}$ such that

$$\pi_{a,k} = \begin{cases} 1, & \text{if } \gamma_a = 1 \text{ and } y_{a,k} = 1, \\ 0, & \text{else} \end{cases}$$

The necessary constraints for this are

$$\pi_{a,k} \leq \gamma_a \quad \forall k \in K,\ \forall a \in A, , \tag{5.4.2a}$$
$$\pi_{a,k} \leq y_{a,k} \quad \forall k \in K,\ \forall a \in A, , \tag{5.4.2b}$$
$$\pi_{a,k} \geq \gamma_a + y_{a,k} - 1 \quad \forall k \in K,\ \forall a \in A, , \tag{5.4.2c}$$
$$\pi_{a,k} \geq 0 \quad \forall k \in K,\ \forall a \in A, \tag{5.4.2d}$$

so a valid reformulation of the uncertainty and lower level as a single-level Integer Program is

$$\max_{\gamma, y, \pi} \quad \sum_{k \in K} \sum_{a \in A} r_a y_{a,k} \tag{5.4.3a}$$

$$s.t. \quad \sum_{a \in A} \gamma_a \leq \Gamma \tag{5.4.3b}$$

$$\sum_{a \in \delta^+(v)} y_{a,k} - \sum_{a \in \delta^-(v)} y_{a,k} = 0, \qquad \forall k \in K,\ \forall v \in V \backslash \{s_k, t_k\} \tag{5.4.3c}$$

$$\sum_{a \in \delta^+(v)} y_{a,k} - \sum_{a \in \delta^-(v)} y_{a,k} = 1, \qquad \forall k \in K,\ v = s_k \tag{5.4.3d}$$

$$\sum_{a \in \delta^+(v)} y_{a,k} - \sum_{a \in \delta^-(v)} y_{a,k} = -1, \qquad \forall k \in K,\ v = t_k \tag{5.4.3e}$$

$$y_{a,k} \leq x_a \qquad \forall k \in K,\ \forall a \in A \tag{5.4.3f}$$

$$\sum_{a \in A} c_a y_{a,k} + c_a^+ \pi_{a,k} \ \leq \ \sum_{a \in p} (c_a + c_a^+ \gamma_a) \qquad \forall k \in K,\ \forall p \in P^k(x) \tag{5.4.3g}$$

$$\pi_{a,k} \leq \gamma_a \qquad \forall k \in K,\ \forall a \in A,, \tag{5.4.3h}$$

$$\pi_{a,k} \leq y_{a,k} \qquad \forall k \in K,\ \forall a \in A,, \tag{5.4.3i}$$

$$\pi_{a,k} \geq \gamma_a + y_{a,k} - 1 \qquad \forall k \in K,\ \forall a \in A,, \tag{5.4.3j}$$

$$\gamma_a \in \{0, 1\} \qquad \forall a \in A \tag{5.4.3k}$$

$$y_{a,k} \geq 0 \qquad \forall k \in K,\ \forall a \in A \tag{5.4.3l}$$

$$\pi_{a,k} \geq 0 \qquad \forall k \in K,\ \forall a \in A, \tag{5.4.3m}$$

To reduce the time needed to solve the reformulated problem, it is useful to employ a cutting-plane approach where only as few path-constraints as necessary are added to the model. If the constraint for a relatively short path is already fullfilled, the constraints for most longer paths are made redundant. More formally, for some path p define lower and upper bounds for the path cost via

$$\begin{aligned} c_{\min}(p) &\coloneqq \sum_{a \in p} c_a, \\ c_{\max}(p) &\coloneqq \max\{c_{\min}(p) + \sum_{a \in T'} c_a^+ \mid T' \subseteq p, |T| \leq \Gamma\} \end{aligned}$$

Then, we have the following property:

Lemma 5. *Let p_1 and p_2 be two paths in $P^k(x)$ with $c_{\max}(p_1) \leq c_{\min}(p_2)$. Then the constraint in* (5.4.3g) *corresponding to p_1 dominates the constraint corresponding to p_2.*

Proof. It must be proven that any solution (γ, y, π) that fullfills the p_1-constraint also fullfills the p_2-constraint. The lefthand-sides of both constraints are identical, whereas

$$\sum_{a \in p_1} (c_a + c_a^+ \gamma_a) \leq c_{\max}(p_1) \leq c\min(p_2) \leq \sum_{a \in p_2} (c_a + c_a^+ \gamma_a).$$

Therefore, the righthand-side of the p_1-constraint is always a lower bound for the righthand-side of the p_2-constraint. □

In other words, any path p with $c_{\min}(p) > \min\{c_{\max}(p') | p' \in P^k(x)\}$ is not important in terms of constraints no matter which arcs have an increased cost.
The resulting idea is as follows: Initialize the reformulation without the constraints given in (5.4.3g). Then, for an optimal solution (γ_0, y_0, π_0) check if the y-variables correspond to cost-minimal paths ("parameterized" by γ). If they do, an optimal solution for the entire reformulation is found. If not, for some commodity k find a path p_k with less cost than the path given by y and add the corresponding constraint.

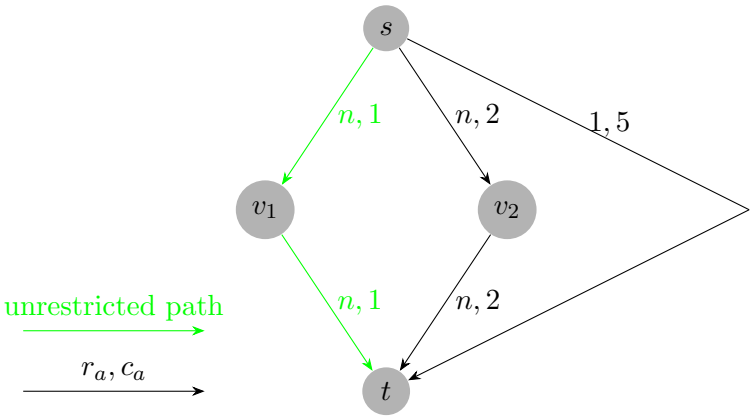

Figure 5.1: If no arcs are removed, the commodity will be transported along the green arcs. The upper level heuristic focuses on removing green arcs. This results in a risk of $2n$, while an optimal solution results in a risk of 1.

The previous reformulation of the uncertainty and lower level assume the leader decision, i.e. the variable x, is fixed. Considering the reformulation is

no longer totally unimodular and the number of constraints due to (5.4.3g) is exponential, it is not simple to find another single-level reformulation that includes both the leader level formulation and the previously archieved reformulation. Instead, heurisitcs can be employed to find a solution that is, if not optimal, then at least reasonably good. However, no fixed approximation factor can be guranteed (as shown in figure 5.1).
The heuristics will be built on the lemma 1 which states that any impactful arc choice of the leader must include at least one arc that would be chosen by the follower in the unrestricted street network. A viable heuristic approach would be the calculation of the set of all streets favoured by the follower $A* \coloneqq \{a \in A | y_{a,k} = 1,\ y \in OPT(1)\}$, where $OPT(1)$ is the set of optimal solutions for formulation 5.3.1 where all arcs are available, i.e. $x_a = 1$ for all $a \in A$.
From this set $A*$, an arc-subset $T \subseteq A* \cap B$ with $|T| \leq b$ can be chosen randomly and removed from the arc of available sets. Afterwards, the reformulation 5.4.3 can be solved on this restricted graph. Depending on the available computation time, this process can be repeated for different choices of T. It should be noted that the removal of arcs can actually increase the resulting risk when compared to the unrestricted case, e.g. if $r_a = c_a$ for all $a \in A$.
This approach can be further refined by calculating the actual risk of an arc based on the follower choice y in the unrestricted case via

$$r'_a(y) \coloneqq \sum_{k \in K} r_a y_{a,k}$$

and for some $m \in \mathbb{N}$, consider only the m arcs of $A*$ with the highest actual risk. Since this reduces $A*$ to a smaller set, it becomes feasible to test all possible choices of up to b arcs instead of an arbitrarily large number of repetitions. The improvement due to this extra step will be tested in the following chapters.

Chapter 6

Implementation

In this chapter, an implementation of the previously introduced formulation is given as proof of concept. The subsequent chapter analyzes the findings of the test routine described below. For this, it is important to detail the hard- and software used as well as the origins of the instances. Additionally an overview of the code and the process of testing is given.
The implementation was written for Python 3.12.0 (see [24]) and utilizes Gurobi 11.0.3 (see [14]). Other libraries used include NetworkX 3.4.2 (see [15]) to handle graphs and the timeit module (part of Python, see [3]) to obtain measurements of computation time. The testing was done using an AMD Ryzen 5 3600.

6.1 Instances

The main component of any HTP-instance is a directed graph $G = (V, A)$. In this graph, every arc a needs a risk value r_a, a cost value c_a and, in case of arc cost uncertainty, a cost increase value c_a^+. Furthermore, a set of commodities $\{(s_k, t_k) \mid k \in K\}$ is needed. Last but not least a leader budget b and an uncertainty budget Γ have be be provided.
The first part of the testing routine is the comparison of the algorithm when used for random graphs (where every arc has a chance of p to be part of the graph) and when used for city street network graphs. While the random graphs were generated by the RandomGraphGen-function (see following section), the city graph instances were taken from [23]. To test the implementation on city graphs of different sizes, instances [21], [8], [18], [19] and [2] (with a node number between 20 and 1000) were utilized. To better show the development of computation time and objective value on a fixed instance, the city graph for "Berlin Prenzlauer Berg" [19] (containing 352 nodes and 749 arcs) was used as standard.
The city graph instances given (more precisely, the "*_net.tntp*"-files) consist

```
<NUMBER OF ZONES> 38
<NUMBER OF NODES> 352
<FIRST THRU NODE> 39
<NUMBER OF LINKS> 749
<ORIGINAL HEADER>~      Init node       Term node       Capacity
<END OF METADATA>


~       init_node       term_node       capacity        length  f
        1       183     999999.0000000000       0.0000000000
        1       294     999999.0000000000       0.0000000000
        1       296     999999.0000000000       0.0000000000
        1       297     999999.0000000000       0.0000000000
        2       187     999999.0000000000       0.0000000000
        2       242     999999.0000000000       0.0000000000
        2       243     999999.0000000000       0.0000000000
        2       245     999999.0000000000       0.0000000000
        3       200     999999.0000000000       0.0000000000
        3       235     999999.0000000000       0.0000000000
        3       318     999999.0000000000       0.0000000000
```

Figure 6.1: Screenshot of the Header of an Instance File. Not All Arcs and Arc Attributes Visible.

of a header listing the number of nodes and arcs, followed by a list of the arcs themselves. While the data set contains several more graph and arc attributes such as capacity and length, the fact that the attributes are either uniform, 0 or practically unlimited for a large subset of the arcs makes them unsuited to be used in the testing. See 6.1 for an example of an instance.
Instead of using the attributes given in the files, each arc risk, cost and cost increase was seperately set to a randomly chosen integer between 1 and 25 (changing this range is part of the later testing). Equally, for every commodity the source and target node were spread randomly across the available nodes. It should be noted that while the existence of at least one path between source and target was ensured, it was possible for a node to serve as source or target for multiple commodities simultanously. The number of commodities was set to a standard value of 5 across all instances, while the leader budget was set to a standard of 3 and the uncertainty budget was set to 5 unless noted otherwise.

6.2 Code Overview - Preliminary Methods

The main method of the implementation requires a directed graph as input and solves the corresponding uncertainty subproblem. The rest of the methods used can be roughly organized into preprocessing (e.g. reading input files and heuristics for the leader level) and analyzing (comparison of

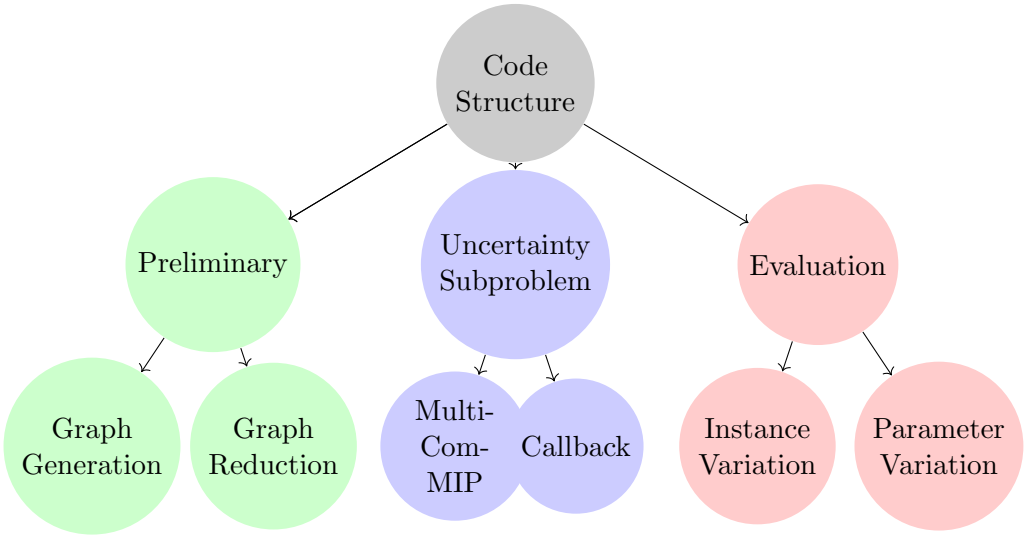

Figure 6.2: Classification of the Implementation Methods

calculation times and objective values for variation of several parameters).
The required directed graphs are created using either the *readInput*-function or the *RandomGraphGen*-function. The former takes as input the filepath of the file to read and optionally the number of commodities to create and the range in which random integers are chosen. It reads the given file in the format outlined in the previous subsection and creates a directed graph accordingly. Missing values such as arc risks are set to random integers in the given range (standard: 25) and commodities (standard number: 5) are spread afterwards. While doing this, it is ensured that no two commodities are identical, that for every commodity there is a path between source and target node and that source and target for a given commodity are not identical.
In contrast, the latter function *RandomGraphGen* needs no filepath, but instead takes the required node number and arc density, i.e. the desired average degree of a node, as input. Similar to above, the commodity number and random range can be given as optional input. It utilizes the *erdos_renyi_graph*-function (see NetworkX-documentation [15]) and, similar to *readInput*, randomly assigns the remaining values in the given range and places the desired number of commodities.
The possible decisions made by the leader regarding the closing of certain arcs are handled by the *GraphReduceRandomly*-function or the *GraphReduceMostRisky*-function. As outlined in 4, it is not possible to find the optimal leader decision in a reasonably short timeframe, which leads to heuristics being employed instead. Both functions take an unrestricted graph, a leader budget (standard: 3) and a desired output size as input. Both produce a

set of graphs where every graph of the set can be obtained by deleting a number of arcs in the input graph which is bounded by the leader budget. The main difference between the functions is in the ways the deleted arcs are chosen.
The *GraphReduceRandomly*-function starts with solving the uncertainty-subproblem for the unrestricted graph. It calculates the set of all arcs that are used for routes according to this solution. It then creates a copy of the input graph and removes a number of arcs out of the set equal to the leader budget from this copy. This process of copy creation and arc deletion is repeated until the desired number of reduced graphs (standard: 10) is reached.
In contrast, the *GraphReduceMostRisky*-function executes an additional step after finding the set of all arcs used for routes. For every arc, the effective risk, i.e. the arc risk multiplied by the number of commodities transported along this arc, is calculated. Afterwards, only those arcs with the highest effective risk (standard: highest 5) are considered for removal. The set of reduced graphs that is returned afterwards contains every possible choice in this smaller arc set, resulting in

$$\binom{5}{0} + \binom{5}{1} + \binom{5}{2} + \binom{5}{3} = 26$$

graphs.

6.3 Code Overview: Solving the Uncertainty Subproblem

The main method of the implementation, *MultiCommodityMip*, recieves as input a graph and optionally an uncertainty budget (standard: 5) and solves the CUS for this instance. Before doing anything else, the program asserts the input is indeed a directed graph with at least one arc having a set risk value, a cost value and a cost increase value (not necessarily the same arc). For every arc not having a set value for one of the three attributes, that value is assumed to be 0 by default. Another requirement is the existence of at least one commodity in the graph.
Assuming the given graph is indeed valid, it is then modified to include the target-source-arcs as outlined in formulation 5.1.2. This means for every commodity the arc from target to source is added if it does not yet exist. To ensure the arcs added this way are not used in any simple minimum-cost path, the cost of the most expensive arc is multiplied by the number of nodes and then given as cost value to the newly created arcs.
This being done, a Gurobi model is created. After initializing the binary follower-variables $\{y_{a,k} \in \{0,1\} | a \in A, k \in K\}$, uncertainty variables $\{\gamma_a \in \{0,1\} | a \in A\}$ and auxiliary variables $\{\pi_{a,k} \in \{0,1\} | a \in A, k \in K\}$,

the objective function (5.4.3a) detailed in the previous section is added. Afterwards, the constraints (5.4.3c), (5.4.3d) and (5.4.3e) ensuring the paths are chosen and the constraint (5.4.3b) limiting the number of arc cost increases are introduced to the model. Constraints (5.4.3h), (5.4.3i) and (5.4.3j) gurantee that the π-variables work as intended. As mentioned in chapter 5, the constraints enforcing that the paths chosen by y are cost-minimal are not added yet. Instead they are treated as lazy constraints and added only as needed during the *MultiCommodityCallback*-part of the optimization process which is started after the other constraints are in place.
For a given integer solution, the *MultiCommodityCallback* checks whether the paths chosen by y are cost-minimal. It consists of two parts: During the first part, a directed graph is constructed using the same vertices and arcs on which the model is based. The arcs are then assigned weights based on γ. It uses either the normal arc cost c_a if $\gamma_a = 0$ or the increased arc cost $()c_a + c_a^+$) if $\gamma_a = 1$. During the second part, for every commodity k the minimal weight of a source-target-path is calculated using the *shortest_path _length*-method (see NetworkX-documentation [15]). If the weight of the arcs chosen by y and π for k, which is

$$\sum_{\substack{a \in A, \\ a \neq (t_k, s_k)}} c_a \cdot y_{a,k} \; + \; c_a^+ \cdot \pi_a$$

exceeds this minimum weight, a minimum-weight path is constructed and its corresponding constraint (5.4.3g) is added to the model.
Only when the path of every commodity is of minimum cost is the model calculation finished.

6.4 Code Overview: Test Routine

The remaining methods of the implementation serve analytic purposes and generate the data which the next chapter is based on. Their structure is relatively similar in that they measure the objective value and computation time needed for a set of graphs. Time measurement is done with the help of the *Timer*-class provided by the timeit-library. Note that the time required to generate the graphs is not included in the measurement.
While the *RandomGraphResults-* and *CityGraphResults*-method compare graphs of varying sizes, i.e. varying node and arc number, while using the standard values for all parameters, the other methods analyzing the effects of parameter variation use the city graph instance [19]. The *RandomGraphResults*-method generates random graphs with varying node numbers (here: $10, 20, \ldots, 100$ nodes) and analyzes the time needed and resulting risk from a run of the *MultiCommodityMip*-method. The *CityGraphResults*-method does the same for given city networks of various size (here: 24, 74, 224, 352

and 1020 nodes). Due to the small size of several of these instances the final time is the amount needed for 5 repetitions.
The methods *ReduceRandomlyResults* and *ReduceMostRiskyResults* serve to analyze the implemented heuristics for the leader decision on which arcs are to be closed. They take a list of input sizes for the mentioned heuristic functions and record the computation times and objective values of the analyzed methods.
In contrast, the methods *UncScalingResults*, *ComScalingResults* and *RandomRangeScalingResults* do not transform the graph, but analyze the effects of different values for the uncertainty budget, number of commodities and range of randomly assigned values, respectively. All three functions document both computation times and objective values. They take as input a set of the different values the parameter in question should assume.

Chapter 7

Computational Results

In this chapter, the results of the implementation will be presented and analyzed. This is done in three parts: Firstly the CUS will be solved on instances of varying size. Secondly the effect of decreasing and increasing the number of commodities and the uncertainty budget for a fixed instance of the CUS will be considered. Thirdly the previously given heuristics for solving the leader level will be tested. In all cases, both computation time and objective value in dependence on the parameter in question will be depicted.

7.1 Instance Variation

In figures 7.1 and 7.2, the computation time needed for solving the CUS-problem on RandomGraph- and CityGraph-Instances of varying size is shown (for different segements of the x-axis). Note that the graphs use a logarithmic scale. While overall, a exponential increase in time can be observed in both types of instances, some instances with higher node number actually require less time than instances with less nodes.

In contrast, figures 7.3 and 7.4 show the risk values of an optimal solution to these instances (using a linear scale). Here, the curiosity of a lower overall risk in a larger graph is even more pronounced. As these instances have a constant number of commodities, it suggests the average path length does in fact decrease. It is also an indication that especially for the RandomGraph-Instances the addition of nodes is more likely to create additional shortcuts than to create additional distance. This seems less pronounced for the more planar city graphs.

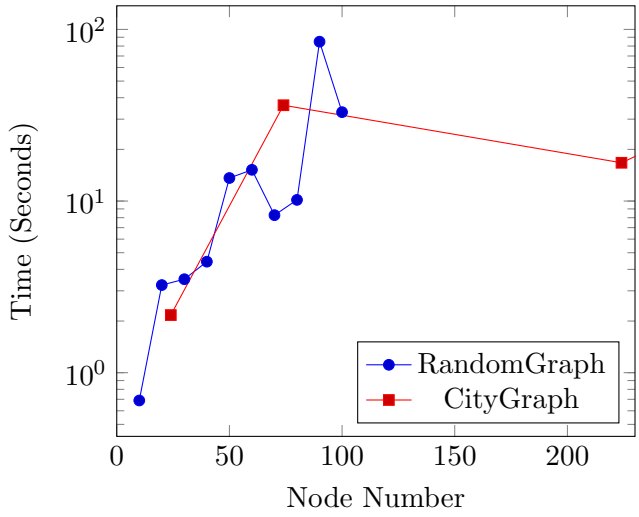

Figure 7.1: Computation Time for RandomGraph- and CityGraph-Instances between 10 and 230 Nodes

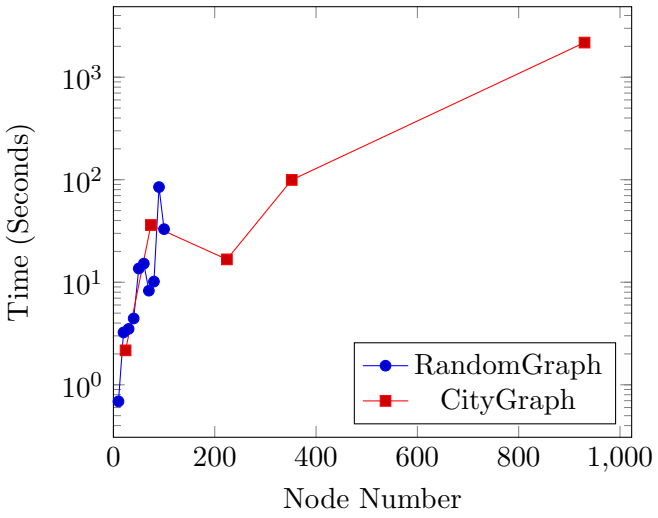

Figure 7.2: Computation Time for RandomGraph- and CityGraph-Instances between 10 and 1000 Nodes

7.2 Commodity and Uncertainty Budget Variation

In the following cases, the instance itself will be fixed to a CityGraph-Instance with 352 nodes. Unless noted otherwise, the number of commodities and uncertainty budget will be set to 5 both, which results in a compu-

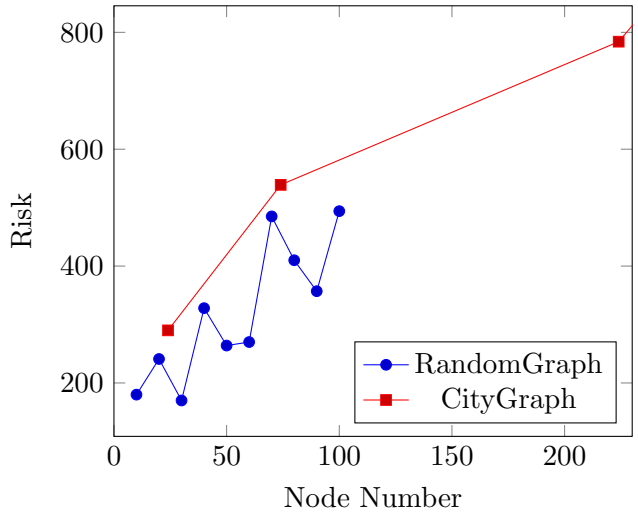

Figure 7.3: Objective Value for RandomGraph- and CityGraph-Instances between 10 and 230 Nodes

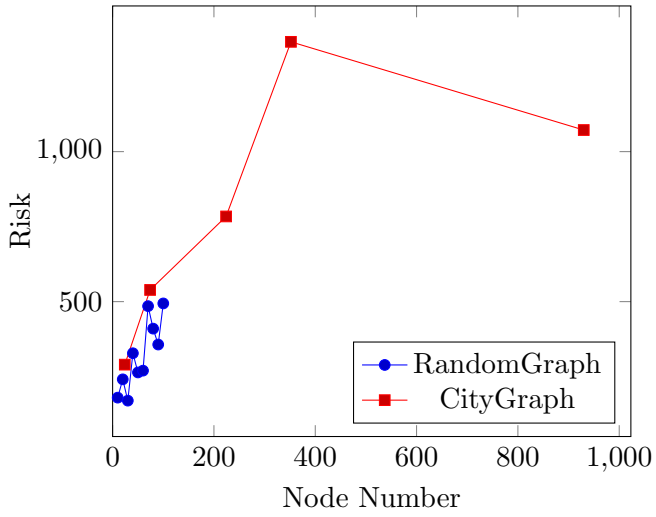

Figure 7.4: Objective Value for RandomGraph- and CityGraph-Instances between 10 and 1000 Nodes

tation time of 18.8 seconds and a overall risk of 1366 if nothing is changed from its standard value.
Figures 7.5 and 7.6 show the time needed (on a logarithmic scale) and the resulting risk (on a linear scale) for different amounts of commodities. While the resulting risk is very much linear, the necessary computation time is superlinear. This observation supports the proposition that even if additional

commodity generate an equal amount of risk, the uncertainty level of the CUS can not treat the commodities like unrelated problems. The cause of the exceptionally high computation time for 25 commodities, being more than twice as high as the time needed for 30 commodities, is unclear.

Figures 7.7 and 7.8 show the impact of variation in the uncertainty budget

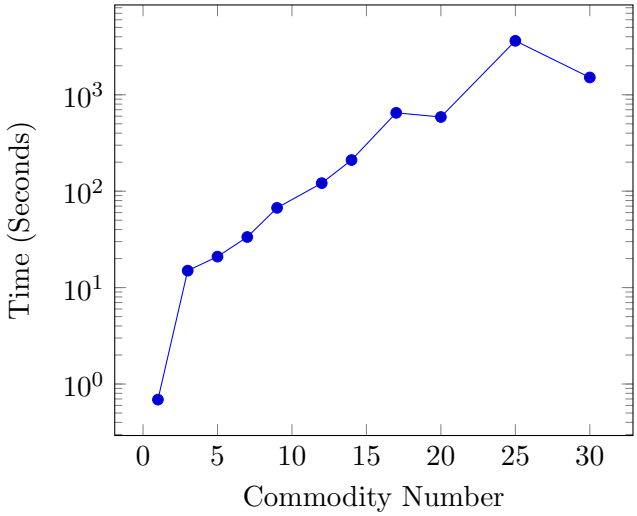

Figure 7.5: Relation Between Number of Commodities and Computation Time

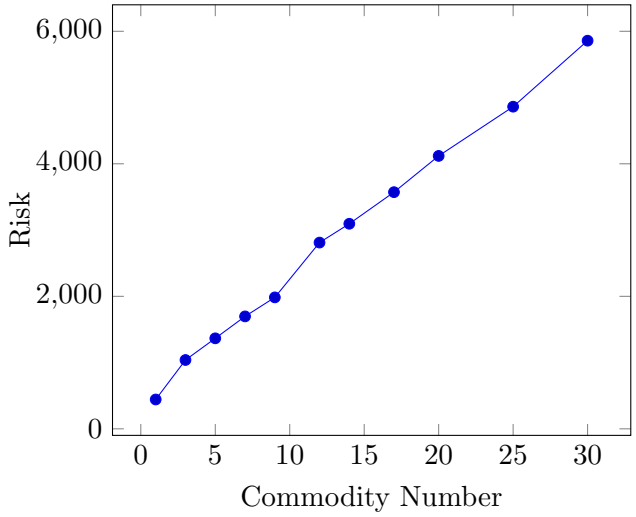

Figure 7.6: Relation Between Number of Commodities and Objective Value

b (the case $b = 0$ represents the non-robust lower level). The first figure (us-

ing a logarithmic scale) shows the same exponential growth in computation time, however in a more explosive manner - e.g. the mark of 10 minutes is surpassed for a uncertainty budget of 11 already while even for a comodity number of 20 this is not the case.
The second figure displays a less-than-linear increase in risk for higher uncertainty budgets. This might be explained by the fact that an increase in uncertainty does not necessarily result in different paths, if the cost difference between paths is too high for the arc cost increases to surpass it. Furthermore, an optimal solution for an unlimited uncertainty budget will not require all arcs to have increased cost, which means the (monotonic) function depicted in figure 7.8 will become constant for a uncertainty budget far less than $|A|$.

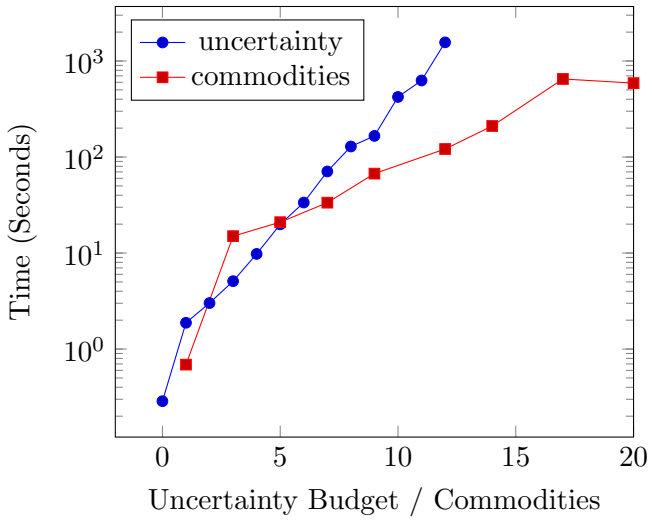

Figure 7.7: Relation Between Uncertainty Budget/Commodities and Computation Time

Since the input files do not cover all parameters necessary for the instances, missing (integer) values are chosen randomly in a range from 1 to 25. Figures 7.9 and 7.10 show the effect of changes in the range. It can be seen that the standard of 25 actually results in the longest computation time. Apart from the expected minimum at 1 (meaning uniform values), it is unclear which computation time should be expected for larger ranges in the choosing of random values. Less unexpected is the fact that the resulting risk scales linearly with the range as the expected value of the arc risks is exactly half the value of the upper range bound.

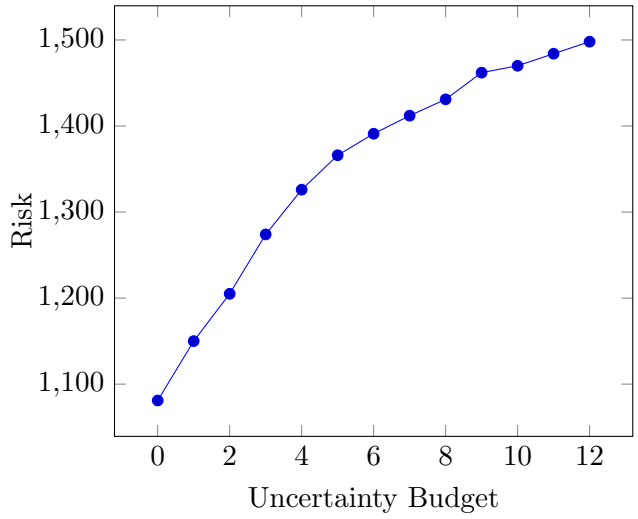

Figure 7.8: Relation Between Uncertainty Budget and Objective Value

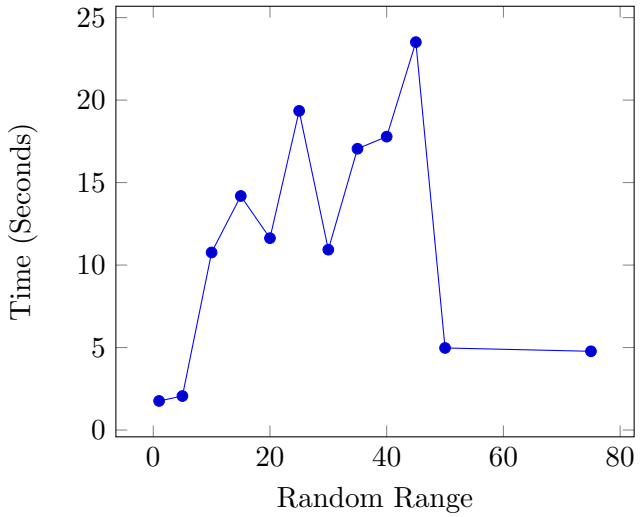

Figure 7.9: Relation Between Range of Random Values and Computation Time

7.3 Upper Level Heuristics

The previously considered results dealt with the CUS on an unrestricted graph. In this section however, the two previously discussed heuristics for the leader level of the HTP are analyzed. Both are based on the arcs used for transport in the unrestricted case. While the RandomlyReduce-heuristic

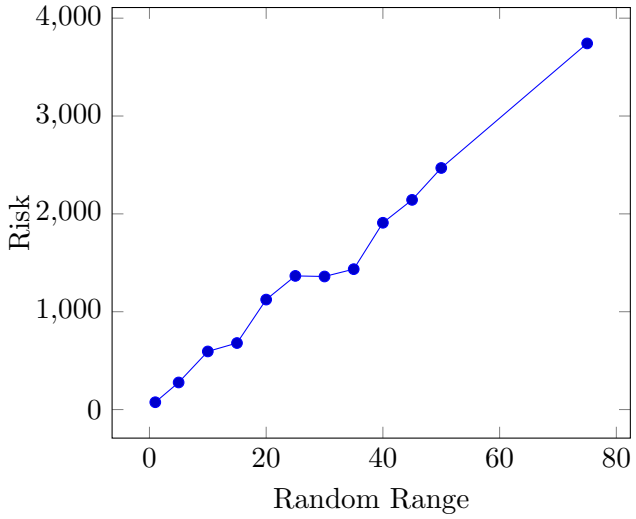

Figure 7.10: Relation Between Range of Random Values and Objective Value

chooses among all arcs and picks the best case among a varying number of choices, the ReduceMostRisky additionally finds a varying number of arcs with the most risk and considers all possible choices among this smaller subset. The actual leader budget of 3 remains unchanged.
This means, a size of k for the RandomlyReduce-heuristic represents a comparison of k choices, whereas for MostRisky-Reduce, this represents $\sum_{i\in\{1,2,3\}} \binom{k}{i}$ cases. For example a size of 8 for RandomlyReduce is equal to a size of 3 for ReduceMostRisky, while 26 for the former is equal to 5 for the latter.
Figures 7.11 and 7.12 show the computation times needed. The results show the expected behaviour, since doubling the repetitions in ReduceRandomly means doubling the work, while the number of cases to consider for ReduceMostRisky has the growth rate of a cubic function.

The actual effectiveness of both heurisitics is depicted in figures 7.13 and 7.14. Given a risk of 1366 in the unrestricted graph, they reduce the risk by up to 52 and 210 respectively, which is 3% and 15% in relative numbers. This shows that even though ReduceMostRisky requires twice as much computation time it gives noticably better solutions.
Another observation is that the rate of improvement sharply declines after the first repetitions/range values. For higher values, the risk is actually constant, meaning the additional arc sets considered for removal are not better than those considered before. While the heuristics lead to noticeable improved solutions, it is doubtful that further increases of the repetitions/range

are worth the additional computation time.

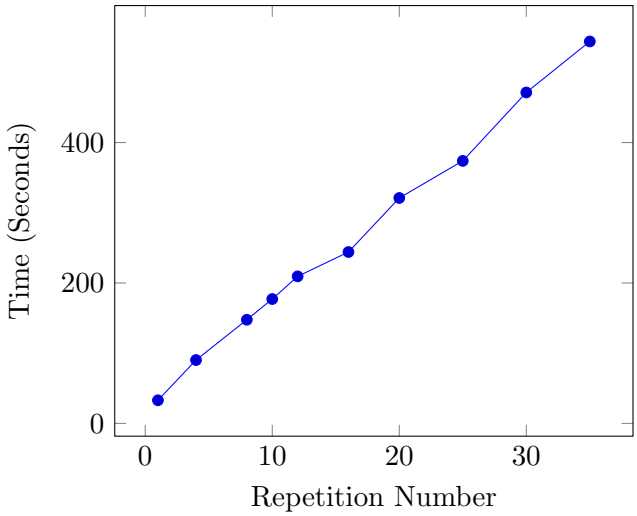

Figure 7.11: Relation Between Repetitions During ReduceRandomly-Heuristic and Computation Time

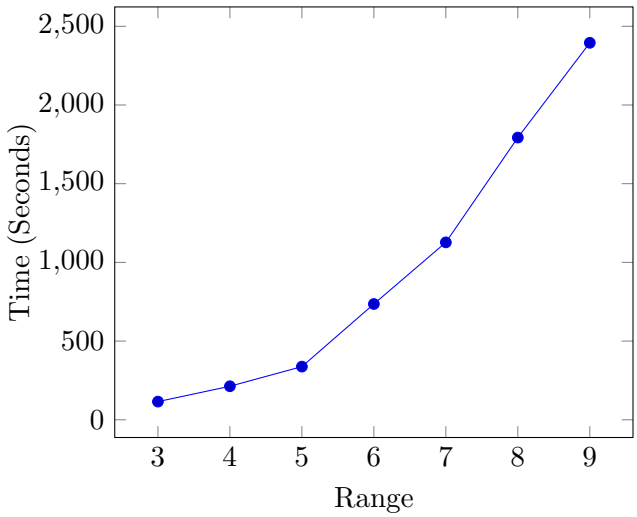

Figure 7.12: Relation Between Range during MostRisky-Heuristic and Computation Time

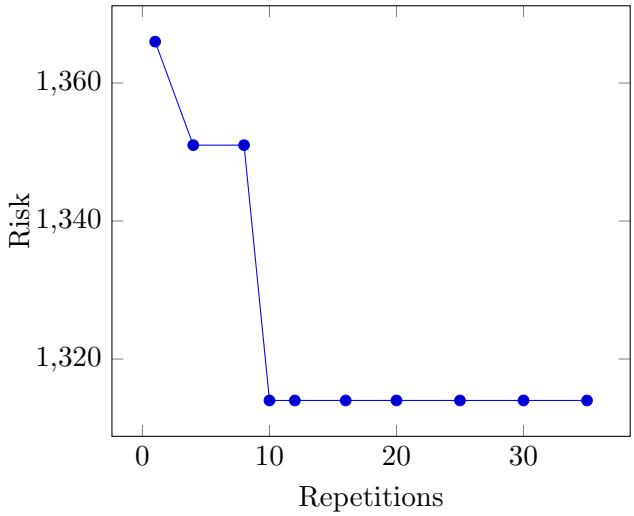

Figure 7.13: Relation Between Repetitions During ReduceRandomly-Heurisitc and Objective Value

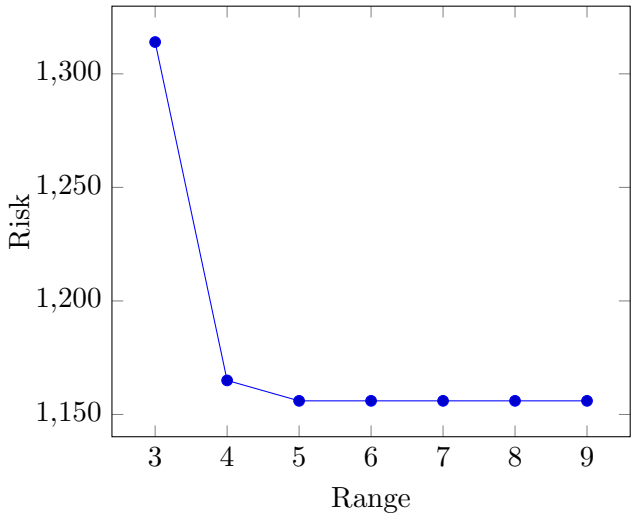

Figure 7.14: Relation Between Range During MostRisky-Heuristic and Objective Value

Chapter 8

Conclusion

In this work, the basic HTP as a bilevel network design problem was modified to a robust version that allows for uncertainty in the arc cost function of the network. Furthermore, it was shown that even finding the worst-case realization of the uncertainty is NP-hard. The robust HTP was then formulated as a multilevel MIP and solution strategies for both the CUS and the robust problem were presented using a single-level reformulation and heuristics. These strategies were implemented and tested in a computational study across several test instances, which also analyzed the influence of several instance parameters. It was shown that the solution strategies, while effective, are also limited in terms of impact and time efficiency.
The results of this work can be expanded on in multiple ways. A natural continuation of the computational study would be testing on further instances derived from real-world scenarios. Another interesting avenue would be the comparison to the approaches of other works regarding the non-robust problem version.
Other research could be focused on the improvement of the heuristics used in this work, which are functional but still unrefined. Both the field of general heuristics and the specifics of the HTP offer many ways of developing alternative solution methods.
And finally, the HTP itself as presented in this work lacks some aspects which are included in some of the other works dealing with Hazmat-Transportation, such as commodity demands or commodity-specific arc risks and arc prohibitions. A more radical change of perspective might even consider other classes of problems to model the risk that arises from the transport of hazardrous materials.

Bibliography

[1] Edoardo Amaldi, Maurizio Bruglieri, and Bernard Fortz. "On the Hazmat Transport Network Design Problem". In: *Network Optimization*. Ed. by Julia Pahl, Torsten Reiners, and Stefan Voß. Vol. 6701. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 327–338. ISBN: 978-3-642-21526-1. DOI: 10.1007/978-3-642-21527-8{\textunderscore}38.

[2] *Barcelona Network*. URL: https://github.com/bstabler/TransportationNetworks/blob/master/Barcelona/Barcelona_net.tntp.

[3] David M. Beazley and Brian K. Jones. *Python cookbook: Recipes for mastering Python 3*. Third edition. Sebastopol, CA: O'Reilly, 2013. ISBN: 9781449357351. URL: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2205818.

[4] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadij S. Nemirovskij. *Robust optimization*. Princeton series in applied mathematics. Princeton, NJ and Oxford: Princeton University Press, 2009. ISBN: 9780691143682.

[5] Paola Cappanera and Maddalena Nonato. "The Gateway Location Problem: A Cost Oriented Analysis of a New Risk Mitigation Strategy in Hazmat Transportation". In: *Procedia - Social and Behavioral Sciences* 111 (2014), pp. 918–926. ISSN: 18770428. DOI: 10.1016/j.sbspro.2014.01.126.

[6] Stephan Dempe et al. *Bilevel Programming Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. ISBN: 978-3-662-45826-6. DOI: 10.1007/978-3-662-45827-3.

[7] Bundesministerium für Digitales und Verkehr. *Verordnung über die innerstaatliche und grenzüberschreitende Beförderung gefährlicher Güter auf der Straße, mit Eisenbahnen und auf Binnengewässern *) (Gefahrgutverordnung Straße, Eisenbahn und Binnenschifffahrt - GGVSEB)*. 2023. URL: https://www.gesetze-im-internet.de/ggvseb/BJNR138900009.html (visited on 12/12/2024).

[8] *Eastern-Massachusetts Network*. URL: https://github.com/bstabler/TransportationNetworks/blob/master/Eastern-Massachusetts/EMA_net.tntp.

[9] Erhan Erkut and Osman Alp. "Designing a road network for hazardous materials shipments". In: *Computers & operations research* 34.5 (2007), pp. 1389–1405. ISSN: 0305-0548. DOI: 10.1016/j.cor.2005.06.007.

[10] Erhan Erkut and Fatma Gzara. "Solving the hazmat transport network design problem". In: *Computers & operations research* 35.7 (2008), pp. 2234–2247. ISSN: 0305-0548. DOI: 10.1016/j.cor.2006.10.022.

[11] Erhan Erkut and Vedat Verter. "Modeling of Transport Risk for Hazardous Materials". In: *Operations Research* 46.5 (1998), pp. 625–642. ISSN: 0030-364X. DOI: 10.1287/opre.46.5.625.

[12] Economic Commission for Europe. *Agreement Concerning the International Carriage of Dangerous Goods by Road (ADR): Applicable As from 1 January 2023*. 1st ed. European Agreement Concerning the International Carriage of Dangerous Goods by Road (ADR) Series. Bloomfield: United Nations Publications, 2022. ISBN: 9789211392111. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=30878412.

[13] Michael R. Garey and David S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. A series of books in the mathematical sciences. New York, NY: W. H. Freeman and Company, 1979. ISBN: 0-7167-1045-5.

[14] L. L.C. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2024. URL: https://www.gurobi.com.

[15] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gael Varoquaux, Travis Vaught, and Jarrod Millman. Proceedings of the Python in Science Conference. SciPy, 2008, pp. 11–15. DOI: 10.25080/TCWV9851.

[16] G. Heilporn et al. "NEW FORMULATIONS AND VALID INEQUALITIES FOR THE TOLL SETTING PROBLEM". In: *IFAC Proceedings Volumes* 39.3 (2006), pp. 431–436. ISSN: 14746670. DOI: 10.3182/20060517-3-FR-2903.00228.

[17] Bahar Y. Kara and Vedat Verter. "Designing a Road Network for Hazardous Materials Transportation". In: *Transportation Science* 38.2 (2004), pp. 188–196. ISSN: 0041-1655. DOI: 10.1287/trsc.1030.0065.

[18] Andreas Schulz Rolf Möhring. *Berlin-Friedrichshain Network*. URL: https://github.com/bstabler/TransportationNetworks/blob/master/Berlin-Friedrichshain/friedrichshain-center_net.tntp.

[19] Andreas Schulz Rolf Möhring. *Berlin-PrenzlauerBerg-Center Network*. URL: https://github.com/bstabler/TransportationNetworks/blob/master/Berlin-Prenzlauerberg-Center/berlin-prenzlauerberg-center_net.tntp.

[20] Saeed Shakeri Nezhad. *A scenario-based hazardous material network design problem with emergency response and toll policy*. 2023. DOI: 10.48336/AHXW-4960.

[21] *SiouxFalls Network*. URL: https://github.com/bstabler/TransportationNetworks/blob/master/SiouxFalls/SiouxFalls_net.tntp.

[22] Longsheng Sun, Mark H. Karwan, and Changhyun Kwon. "Robust Hazmat Network Design Problems Considering Risk Uncertainty". In: *Transportation Science* 50.4 (2016), pp. 1188–1203. ISSN: 0041-1655. DOI: 10.1287/trsc.2015.0645.

[23] Transportation Networks for Research Core Team. *Transportation Networks for Research*. URL: https://github.com/bstabler/TransportationNetworks.

[24] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[25] Vedat Verter and Bahar Y. Kara. "A Path-Based Approach for Hazmat Transport Network Design". In: *Management Science* 54.1 (2008), pp. 29–40. ISSN: 0025-1909. DOI: 10.1287/mnsc.1070.0763.

[26] Jinpei Wang, Xuejie Bai, and Yankui Liu. "Globalized robust bilevel optimization model for hazmat transport network design considering reliability". In: *Reliability Engineering & System Safety* 239 (2023), p. 109484. ISSN: 09518320. DOI: 10.1016/j.ress.2023.109484.

[27] Laurence A. Wolsey. *Integer programming*. Second edition. Hoboken, NJ and Chichester, West Sussex: Wiley, 2021. ISBN: 9781119606536. URL: https://zbmath.org/?q=an%3A7267975.

[28] Chunlin Xin et al. "Robust optimization for the hazardous materials transportation network design problem". In: *Journal of Combinatorial Optimization* 30.2 (2015), pp. 320–334. ISSN: 1382-6905. DOI: 10.1007/s10878-014-9751-z.