Integer Linear Programming for Trust-Region Subproblems in Integer Optimal Control with Total Variation Regularization

Dissertation
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Marvin Pascal Severitt

im September 2025

Dissertation

Integer Linear Programming for Trust-Region Subproblems in Integer Optimal Control with Total Variation Regularization

Fakultät für Mathematik
Technische Universität Dortmund

Erstgutachter: Juniorprof. Dr. Paul Manns

Zweitgutachter: Prof. Dr. Marc Pfetsch

Tag der mündlichen Prüfung: 17.12.2025

Abstract

In this work we concern ourselves with integer linear programs we obtain from a uniform discretization of subproblems arising in a trust-region algorithm for integer optimal control problems with total variation regularization. The underlying domain is either one-dimensional or two-dimensional and the control value set is a finite contiguous subset of the integers. We introduce several relaxations which not only provide lower bounds for our integer programming formulation but also allow us to obtain a conditional p-approximation.

We provide NP-hardness results which show that a more general version of the integer program obtained from a non-uniform discretization of the trust-region subproblem is NP-hard by a reduction from the knapsack problem. Furthermore, we conjecture the NP-hardness of the two-dimensional case even with a binary control value set. We support this conjecture by providing a reduction from the minimum bisection problem for grid graphs with an arbitrary number of holes which has been conjectured to be NP-hard for several decades.

We show that the polyhedron of the linear programming relaxation has a very special property. For vertex solutions we prove that non-integer values can only be attained in entry combinations subject to structural restrictions which also enforce that the fractional value attained in each of these entries is identical. This property is reminiscent of the solution for the relaxed knapsack problem in which at most one entry is fractional.

For the one-dimensional case we are able to provide a shortest path approach for the more general case of a non-uniform discretization with a corresponding integer program. This provides a pseudo-polynomial algorithm. For the two-dimensional case we employ an integer programming solver instead. We can use the one-dimensional case to calculate as well as improve good feasible points which allows for a primal heuristic. Furthermore, we supply a simple branching priority derived from the one-dimensional case. Based on the special property of the polyhedron, we derive cutting planes which make use of a connection to graph-based problems as well as the so-called minimum cut ratio. Finally, we extend a decomposition approach for the overall trust-region algorithm.

We validate our approaches with several numerical examples and note significant run time improvements for the one-dimensional and two-dimensional cases. We discuss under which conditions our approaches are especially effective but also argue their limitations.

Zusammenfassung

In dieser Arbeit befassen wir uns mit ganzzahligen linearen Programmen, welche wir durch eine uniforme Diskretisierung von Subproblemen erhalten, die aus einem Trust-Region Verfahren für ganzzahlige Optimalsteuerungsprobleme mit Totalvariationsbestrafungsterm stammen. Die zugrundeliegende Domäne ist entweder eindimensional oder zweidimensional und die Menge der Steuerungswerte ist eine endliche Teilmenge der ganzen Zahlen mit aufeinanderfolgenden Werten. Wir führen mehrere Relaxierungen ein, welche uns nicht nur untere Schranken liefern, sondern auch eine bedingte p-Approximation.

Wir geben mehrere NP-Schwere Resultate an, welche zeigen, dass eine allgemeinere Form der ganzzahligen Programme, welche wir durch eine nicht uniforme Diskretisierung der Trust-Region Subprobleme erhalten, NP-schwer ist, indem wir eine Reduktion vom Rucksackproblem angeben. Weiterhin stellen wir die Vermutung auf, dass das zweidimensionale Problem selbst mit einer binären Steuerung NP-schwer ist. Wir untermauern die Vermutung, indem wir eine Reduktion vom minimalen Bisektionsproblem auf Gittergraphen mit einer beliebigen Anzahl von Löchern aufzeigen, wobei für dieses Problem die NP-Schwere seit mehreren Jahrzehnten vermutet wird.

Wir zeigen, dass das zu der LP-Relaxierung gehörige Polyeder eine besondere Struktur aufweist. Für alle Ecklösungen zeigen wir, dass eine strukturelle Einschränkung gilt, sodass nicht ganzzahlige Werte nur für bestimmte Eintragskombinationen auftreten können und alle nicht ganzzahligen Einträge schon den gleichen Wert aufweisen. Diese Eigenschaft erinnert an das relaxierte Rucksackproblem, bei dem die Lösung höchstens einen fraktionalen Eintrag enthält.

Für den eindimensionalen Fall können wir für den allgemeineren Fall einer nicht uniformen Diskretisierung mit entsprechendem ganzzahligen Programm eine Formulierung als kürzeste Wegeproblem angeben. Wir erhalten einen pseudopolynomiellen Algorithmus. Für den zweidimensionalen Fall nutzen wir einen Löser für ganzzahlige Programme. Wir können auf den eindimensionalen Fall zurückgreifen, um gute zulässige Punkte zu errechnen und zu verbessern, sodass wir eine primale Heuristik erhalten. Weiterhin nutzen wir den eindimensionalen Fall um eine Verzweigungspriorität anzugeben. Wir nutzen die besondere Eigenschaft des Polyeders der LP-Relaxierung, um Schnittebenen zu konstruieren, welche die Nähe zu graphenbasierten Problemen und dem sogenannten minimalen Schnittverhältnis ausnutzen. Zu guter Letzt erweitern wir einen Dekompositionsansatz für das Trust-Region Verfahren.

Wir validieren unsere Ansätze mithilfe von mehreren numerischen Experimenten, in welchen wir einen signifikanten Laufzeitgewinn sowohl für den eindimensionalen als auch für den zweidimensionalen Fall feststellen. Wir diskutieren unter welchen Bedingungen unsere Ansätze besonders gut funktionieren und was die Grenzen sind.

Acknowledgement

I would like to thank my supervisor Paul Manns for the opportunity to pursue my PhD, the helpful discussions and plenty of useful advice. I would also like to thank Prof. Marc Pfetsch for being the second referee and providing me the opportunity to present my topic as part of the optimization seminar in Darmstadt. I am grateful for the opportunity of my stay at Argonne and want to thank everyone who made this possible.

I am very grateful for the full support I have received from my parents, Jens and Andrea Severitt. I also want to highlight the dogs, Floki and Nimue.

I want to express my gratitude to my family, my friends (especially GBN), my colleagues and all the other people who have supported me over the years.

Contents

Chapter 1

Introduction

The problem class of (mixed-)integer optimal control problems (IOCP) offers a tool to model a variety of applications. Examples include but are not limited to supply network models [62], network transportation problems in the form of traffic flow [48, 49] or gas flow [52, 55], and topology optimization [57, 67, 68, 97]. They extend the problem class of optimal control problems, which are optimization problems constrained by an ordinary or partial differential equation, by enforcing integrality constraints on the control. This allows the modeling of discrete quantities as they appear in many real world applications. The improvements in modeling power come at the expense of a need for optimization approaches which can account for the integrality constraints and a corresponding additional computational demand. A popular approach to solve IOCPs is the combinatorial integral approximation (CIA) decomposition, see [89]. The approach solves a continuous relaxation first and afterwards calculates an integer-valued approximation. Under suitable assumptions on the partial differential equation, this approach gives optimality guarantees in the form that the difference between the objective values to the relaxed problem and the integer-valued approximation obtained tend to zero as the mesh, for which the solutions are calculated, is refined. The underlying optimality principle alone does not suffice to prevent high oscillation of the control function, see [61], which is detrimental for many real world applications. In turn, it requires the inclusion of switching constraints or total variation constraints in the approximation step which impede the aforementioned optimality guarantee, see [89, 90]. Alternatively, one can include switching costs in the rounding framework in a way which still obtains approximation guarantees with the drawback that high-frequency switching can still occur, see [11–13].

In the final part of the three part series [22–24] a branch-and-bound approach is presented for parabolic optimal control problems with combinatorial switching constraints. The set of feasible controls is a bounded subset of the set of functions with bounded variations. In the first two parts of the series they present a convexification of their problem class and an approach to obtain a solution for the relaxations (outer

approximation algorithm) which is integrated into the branch-and-bound approach. They also study how additional linear inequalities with regards to the optimal control can be added. In total, they are able to obtain an algorithm which produces globally optimal solutions for the problem class.

A different approach is to include a total variation regularization in the problem formulation as done in [66]. A trust-region algorithm for IOCPs with total variation regularization was proposed for a one-dimensional underlying domain, which was later extended to multiple dimensions in [73]. The bulk of the computational demand of the trust-region algorithm stems from the trust-region subproblems of the form

$$\begin{aligned} \min_{d \in L^2(\Omega)} \; & (\nabla F(u), d)_{L^2(\Omega)} + \alpha \mathrm{TV}(u+d) - \alpha \mathrm{TV}(u) \\ \text{s.t.} \; & u(x) + d(x) \in \Xi = \{\xi_1, \dots, \xi_m\} \subset \mathbb{Z} \text{ for a.a. } x \in \Omega, \\ & \|d\|_{L^1(\Omega)} \leq \Delta. \end{aligned}$$

After discretization the problems can be solved as (mixed-)integer linear programs. The subproblems without the trust-region constraint $\|d\|_{L^1(\Omega)} \leq \Delta$ hold strong similarities to other well-studied problems. A related field of study encompasses Markov random fields (MRFs) and the Potts model in the context of vision problems, see [45,84]. For an overview with regards to the Potts model for labeling optimization we refer the reader to [78]. The study of MRFs and study of total variation minimization bear a strong connection, as can for example be seen in [28] in which the relationship between a class of binary MRF models and the total variation minimization for image denoising (ROF model), see [86], is studied. A commonly used tool in the context of MRFs are graph cuts, see for example [19], which is an avenue we will also explore later on.

In this work we will study discretizations of these trust-region subproblems for one-dimensional or two-dimensional domains and, to a lesser degree, the trust-region algorithm itself with the aim of reducing the computational demand of the overall algorithm. Just recently a more general form of the discretized problem with univariate functions has been studied in [104]. They refer to several results presented in this work published in [96] and [74] but pursue a different approach based on the Graver basis, see [50], which allows for the construction of a randomized heuristic algorithm. For this purpose they use the results presented in [103] in which a specialized simplex method for the linear programming relaxation of the integer programming formulation of the trust-region subproblem with $u \equiv 0$ is proposed.

Contribution

We introduce the discretization of the trust-region subproblem and the resulting integer programming formulation. Furthermore, we present the underlying graph structure resulting from the discretization. We derive two different relaxations to the

integer programming formulation apart from the linear programming relaxation. One of the relaxations, a Lagrangian relaxation of the trust-region constraint, will allow us to provide a result about a conditional p-approximation for the trust-region subproblem based on a solution for this Lagrangian relaxation. The other relaxation is based on the idea of the dual decomposition approach. This relaxation provides additional lower bounds for the integer programming solution which are at least as good as the lower bound obtained from the linear programming relaxation. We will show the connection to graph-based problems for the integer programming formulation as well as for the relaxations.

If the underlying domain Ω is one-dimensional, we are able to reformulate the problem as a resource-constrained shortest path problem in which the discretized trust-region constraint acts as the capacity constraint. We are also able to reformulate the problem as a shortest path problem by encoding the capacity in the node construction for the corresponding layered directed acyclic graph. In consequence, the resulting graph size depends on the trust-region radius Δ. We provide two solution approaches to solve the shortest path approach. The first approach uses the topological order induced by the graph construction to obtain a dynamic programming approach to solve the shortest path problem. The second approach uses an A^* algorithm with a consistent heuristic function derived from the Lagrangian relaxation of the trust-region constraint. The dynamic programming approach can be extended to the case that the underlying graph structure is a tree.

For the two-dimensional case we will show the connection to the constrained minimum s-t cut problem and the minimum bisection problem. We provide an NP-hardness result by a reduction from the knapsack problem for a more general case than discussed in this work and conjecture that the problem remains NP-hard even for our case with a binary set Ξ. We support this conjecture with a reduction from the minimum bisection problem for grid graphs with an arbitrary number of holes, which is conjectured to be NP-hard. We give structural results for the polyhedron of the linear programming relaxation. We show that for vertices of the polyhedron it holds that non-integer values only occur for entries corresponding to nodes of a connected subgraph with regards to the underlying graph derived from the discretization, which we refer to as the fractional component. We use these insights to provide a primal heuristic, a branching priority and cutting planes for the integer programming formulation. The cutting planes use the concept of a minimum cut ratio for this fractional component. We extend an existing decomposition approach for the overall trust-region algorithm by using the insights gained regarding the linear programming relaxation. We validate all the provided approaches with several numerical examples and discuss the improvements by the approaches as well as the limitations. In the appendix we provide the theoretical basis for additional theoretical avenues which did not produce satisfying computational results in practice but might encourage future promising research.

Structure of the dissertation

In Chapter 2 we provide the reader with the needed preliminary knowledge for the ideas and results of this work. In Chapter 3 we present the trust-region algorithm from which our integer programs arise after discretization of the subproblems. The integer programming formulation and a multitude of relaxations are given in the following Chapter 4. The problems of concern have connections to several graph-based problems which we detail in Chapter 5. Furthermore, these graph-based problems lay the foundation for a discussion about the NP-hardness of the problem, which follows in Chapter 6. In order to solve the integer programs we discuss the underlying polyhedron of the linear programming relaxation in Chapter 7 and derive a unique property of the polyhedron which will prove essential in deriving approaches to reduce the run time. These algorithmic approaches are given in Chapter 8 for the one-dimensional and two-dimensional case. In the one-dimensional case a graph construction derived in Chapter 5 allows for a reformulation as a shortest path problem. For the two-dimensional case we employ an integer programming solver for which we provide a primal heuristic, a branching priority and cutting planes. The cutting planes are based on the property of the polyhedron derived in Chapter 7. Furthermore, we propose improvements to the decomposition approach from [6] in Chapter 9 to improve the overall trust-region algorithm. To validate our proposed approaches we run several numerical experiments and analyze the results in Chapter 10. Finally, we conclude this thesis with an outlook for further research in Chapter 11. In the Appendix A we have provided for the interested reader some theoretical avenues which did not prove useful in initial computational experiments but might work with some smart improvements and implementations in the future.

Notation For convenience and improved visual clarity we use the short notation $[N] \coloneqq \{1, \dots, N\}$. We introduce the notation $\lfloor x \rceil$ to represent the rounding of x to the nearest integer value. In case of parity, x is rounded up. The notation $\lceil x \rceil$ denotes rounding up to the nearest larger integer value while $\lfloor x \rfloor$ denotes rounding down.

Publications During the study of the dissertation topic three papers have been published with the results obtained during the time of study. The first paper [96] is an extension of the master thesis [95] which deals with the one-dimensional problem formulation. The results were extended to the case of a non-uniform discretization, the presentation was significantly improved and additional numerical results were provided. The proofs were extended to the non-uniform case, the arguments were improved and mistakes were corrected. We give the overhauled presentation of the results and statements but have only included proofs which were not part of [95]. In the cases that the original version of the proofs stem from [95], we instead only refer to their improved version in [96] to draw a clear distinction between the work in [95]

and this dissertation. The second paper [74] discusses the main contributions of this dissertation but does not contain a variety of insights presented in this work. The final paper is [15] and the work therein is not included in this dissertation.

Chapter 2

Preliminary knowledge

In this chapter we introduce the needed concepts which form the basis for this dissertation.

2.1 Graph theory

In this dissertation we will discuss a multitude of combinatorial problems which can be viewed as graph-based problems. Thus, graph theory plays a major role in the following chapters and we will state the necessary preliminary knowledge here. We have consulted [63] and [35] for this subsection but note that there is an extensive amount of literature for the basics of graph theory with slight variations in their definitions.

Definition 2.1. An **(undirected) graph** $G = (V, E)$ consists of a set of nodes/vertices V and a set of edges $E = \{\{v_i, v_j\} \mid v_i, v_j \in V\}$. A graph $\tilde{G} = (\tilde{V}, \tilde{E})$ is called a **subgraph** of G if $\tilde{V} \subset V$ and $\tilde{E} = \{\{v_i, v_j\} \in E \mid v_i, v_j \in \tilde{V}\} \subset E$. A **directed graph** $G = (V, A)$ consists of a set of nodes V and a set of edges $A = \{(v_i, v_j) \mid v_i, v_j \in V\}$ for which each edge $e = (v_i, v_j)$ is assigned an initial node v_i and a terminal node v_j such that e is directed from v_i to v_j.

The following two concepts are introduced for undirected graphs but also apply to the directed case.

Definition 2.2. Two nodes $v, w \in V$ are called **adjacent** if $e = \{v, w\} \in E$. In this case, the nodes are called **incident** to the edge e. Two edges e_1, e_2 are called adjacent if they are incident to the same node.

Definition 2.3. A path P in G is a subgraph $P(\{v_1, \ldots, v_{k+1}\}, \{e_1 = \{v_1, v_2\}, \ldots, e_k = \{v_k, v_{k+1}\}\})$ with $v_i \neq v_j$ for $1 \leq i < j \leq k + 1$. The length k of the path is given by the number of edges.

Definition 2.4. We say an undirected graph $G = (V, E)$ is **connected** if for all $v_i, v_j \in V$ we can find a path from v_i to v_j. A graph is called a **tree** if for all $v_i, v_j \in V$ there exists exactly one path from v_i to v_j.

We will encounter planar and grid graphs which both can be embedded into the plane $\mathbb{R}^2$ with additional properties. We start by introducing planar graphs but need some prior definitions to characterize the embedding.

Definition 2.5 (Definition 2.28 in [63])**.** A **simple Jordan curve** is the image of a continuous injective function $\phi : [0, 1] \to \mathbb{R}^2$; its endpoints are $\phi(0)$ and $\phi(1)$. A **closed Jordan curve** is the image of a continuous function $\phi : [0, 1] \to \mathbb{R}^2$ with $\phi(0) = \phi(1)$ and $\phi(\tau) \neq \phi(\tau')$ for $0 \leq \tau < \tau' < 1$. A **polygonal arc** is a simple Jordan curve which is the union of finitely many intervals (straight line segments). A **polygon** is a closed Jordan curve which is the union of finitely many intervals.

Let $R = \mathbb{R} \setminus J$, where J is the union of finitely many intervals. We define the **connected regions** of R as equivalence classes where two points in R are equivalent if they can be joined by a polygonal arc within R.

Definition 2.6 (Definition 2.29 in [63])**.** A **planar embedding** of a graph $G = (V, E)$ consists of an injective mapping $\psi : V \to \mathbb{R}^2$ and for each $e = \{x, y\} \in E$ a polygonal arc J_e with endpoints $\psi(x)$ and $\psi(y)$, such that for each $e = \{x, y\} \in E$:

$$(J_e \setminus \{\psi(x), \psi(y)\}) \cap \left(\{\psi(v) : v \in V\} \cup \bigcup_{e' \in E \setminus \{e\}} J_{e'} \right) = \emptyset.$$

A graph is called **planar** if it has a planar embedding.

Let G be a planar graph with some fixed planar embedding $\Phi = (\psi, (J_e)_{e \in E})$. After removing the points and polygonal arcs from the plane, the remainder,

$$R \coloneqq \mathbb{R}^2 \setminus \left(\{\psi(v) : v \in V\} \cup \bigcup_{e \in E} J_e \right)$$

splits into open connected regions, called **faces** of Φ.

All faces are bounded except one which is the so-called **outer face (or infinite face)**, see [63] p. 37-39 and [35] p. 97.

Thus, planar graphs are graphs which can be drawn in the plane such that the edges do not intersect. For every planar graph we can also define the dual graph.

Definition 2.7 (Definition 2.41 in [63])**.** Let $G = (V, E)$ be a directed or undirected graph and let $\Phi = (\psi, (J_e)_{e \in E})$ be a planar embedding of G. We define the **planar dual (graph)** G^* whose vertices are the faces of Φ and whose edge set is $\{e^* : e \in E\}$ where e^* connects the faces that are adjacent to J_e.

An important class of planar graphs for the following chapters are grid graphs.

Definition 2.8. The infinite grid (graph) is the graph embedded in $\mathbb{R}^2$ with $V = \mathbb{Z}^2$ and $E = \{e = \{v_i, v_j\} \mid v_i, v_j \in V,\ ||v_i - v_j||_1 = 1\}$. A grid graph is a finite node-induced subgraph of the infinite grid.

We note that the infinite grid is often introduced with $V = \mathbb{N}^2$ instead, see [35], but we have choosen this definition which can be found in many papers, see for example [59] or [98], as this is the definition used in [80] which we need later on. Furthermore, [80] introduces the concept of holes for grid graphs which we state below.

Definition 2.9 (See [80])**.** A hole is a finite connected component of the complement of the grid graph with respect to the infinite grid. A grid graph without holes is called solid.

2.2 Integer programming

We will see in a later chapter that the subproblems we concern ourselves with can be formulated as integer programs. We do not give an overview for all the basic concepts but focus on the most important concepts instead. The reader needs a basic understanding of linear programming, duality, reduced costs, the simplex algorithm, polyhedra, facets, integer programming and the branch-and-bound algorithm.

An important role to decrease the run time of the integer programming solver is to improve the linear programming formulation by adding so-called cutting planes. We first define the integer hull and then introduce cutting planes. Let the linear program

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & x \in P \end{aligned} \tag{2.1}$$

with $P = \{x \in \mathbb{R}^n \mid Ax \geq b, x \geq 0\}$ be given. The set of feasible integer points is given by $\{x \in P \mid x_i \in \mathbb{Z} \text{ for all } i \in \{1, \ldots, n\}\}$.

Definition 2.10 (Chapter 5 in [63])**.** The **integer hull** is the convex hull of the feasible integer points $P_I = \text{conv}(\{x \in P \mid x_i \in \mathbb{Z} \text{ for all } i \in \{1, \ldots, n\}\})$.

We will later on assume rational data. This also guarantees that the integer hull is a polyhedron.

Theorem 2.11 (Theorem 5.1 in [63])**.** *For any rational polyhedron P, its integer hull P_I is a rational polyhedron.*

Proof. See [63] and [76]. □

Thus, one could determine the integer hull in order to solve an integer program on P as a linear program on P_I instead. This tends to demand a lot of effort, which is why cutting planes are used instead to strengthen the linear programming relaxation, see Section 1.2 in [5].

Definition 2.12 (Section 1.2 in [5])**.** A linear inequality $a^T x \leq \alpha$ with $a^T \bar{x} \leq \alpha$ for all $\bar{x} \in P_I$ and $a^T x^* > \alpha$ for an $x^* \in P$ is called a **cutting plane** and we say that it cuts off a part of P but no part of P_I.

We will introduce the cutting planes in Section 8.3.3 and validate in Chapter 10 that this allows for an improved linear programming relaxation and a reduction in the run times of the integer program solver.

Remark 2.13. For the solution process it would also be acceptable to introduce inequalities which cut off feasible integer points which are not optimal. In this disseration we follow the above definition and our cutting planes do not cut off feasible suboptimal integer points.

2.3 Function space and bounded variation

The topic of this dissertation are the integer programs we derive as subproblems from the trust-region algorithm. For a proper contextualization regarding the assumptions and convergence results of the subordinate problem and trust-region algorithm, we provide the reader with a short summary of the necessary preliminary knowledge. This knowledge is not required for the results outside of Chapter 3 and Chapter 9. As a basis for this section we have used [2].

In the following $\Omega \subset \mathbb{R}^d$ denotes an open set with $d \in \{1, 2\}$. We first introduce the concepts of total variation of a function and the space of functions of bounded variation.

Definition 2.14 (Definition 6.102 and Lemma 6.103 in [20], see also p. 116-120 in [2])**.** Let $u \in L^1(\Omega)$. We define the **total variation** of u by

$$\mathrm{TV}(u) \coloneqq \sup \left\{ \int_\Omega u(x) \operatorname{div} \phi(x) dx \;\middle|\; \phi \in C_c^1(\Omega; \mathbb{R}^d),\ \operatorname*{ess\,sup}_{x \in \Omega} ||\phi(x)||_2 \leq 1 \right\}$$

where $\operatorname{div} \phi(x) = \sum_{i=1}^d \frac{\partial}{\partial x_i} \phi_i(x)$, $C_c^1(\Omega; \mathbb{R}^d)$ is the space of continuously differentiable functions $f : \Omega \to \mathbb{R}^d$ with compact support. The **space of functions of bounded variation** is given by

$$\mathrm{BV}(\Omega) \coloneqq \{u \in L^1(\Omega) \mid \mathrm{TV}(u) < \infty\}.$$

We introduce the notations $L^1_\Xi(\Omega) \coloneqq \{u \in L^1(\Omega) \mid u(x) \in \Xi \text{ for a.a. } x \in \Omega\}$ and $\mathrm{BV}_\Xi(\Omega) \coloneqq \mathrm{BV}(\Omega) \cap L^1_\Xi(\Omega)$ for a finite set $\Xi \subset \mathbb{Z}$.

For the convergence results in Chapter 3 we need to introduce the notion of weak-$*$ convergence for functions in $\mathrm{BV}(\Omega)$.

Definition 2.15 (Combination of Definition 3.11 and Proposition 3.13 in [2])**.** Let $\{u_k\}_{k\in\mathbb{N}} \subset \mathrm{BV}(\Omega)$ be a sequence and $u \in \mathrm{BV}(\Omega)$. We say that $\{u_k\}_{k\in\mathbb{N}}$ converges weakly-$*$ in $\mathrm{BV}(\Omega)$ to u for $k \to \infty$, denoted by $u_k \overset{*}{\rightharpoonup} u$, if $\{u_k\}_{k\in\mathbb{N}}$ is bounded in $\mathrm{BV}(\Omega)$ and $u_k \to u$ in $L^1(\Omega)$ for $k \to \infty$.

For a piecewise constant $u \in \mathrm{BV}_\Xi(\Omega)$ we want to characterize the total variation as the sum of the products of the jump heights and the Hausdorff measure of the intersection of two sets of a partition into polyhedra, which we will obtain in the following.

Definition 2.16 (Definition 2.46 in [2])**.** Let $k \in [0,\infty)$ and $E \subset \mathbb{R}^d$. The k **dimensional Hausdorff measure** of E is given by

$$\mathcal{H}^k(E) \coloneqq \lim_{\delta \searrow 0} \mathcal{H}^k_\delta(E)$$

where, for $0 < \delta \leq \infty$, $\mathcal{H}^k_\delta(E)$ is defined by

$$\mathcal{H}^k_\delta(E) \coloneqq \frac{\omega_k}{2^k} \inf\left\{ \sum_{i\in I} \mathrm{diam}(E_i)^k \ : \ \mathrm{diam}(E_i) < \delta, E \subset \bigcup_{i\in I} E_i, I \subset \mathbb{N} \right\}$$

with $\omega_k = \lambda_{\mathbb{R}^k}(B_1^{\mathbb{R}^k}(0))$, $\mathrm{diam}(E_i) = \sup_{x,y\in E_i} ||x-y||$ for the Euclidean norm on $\mathbb{R}^d$ and the convention $\mathrm{diam}(\emptyset) = 0$.

Lemma 2.17 (Special case of Lemma 2.1 (c) in [73])**.** *Let $\sum_{i=1}^N \xi_{E_i}\chi_{E_i} = u \in \mathrm{BV}_\Xi(\Omega)$ with a partition $\{E_1,\dots,E_N\}$ of Ω into polyhedra. Then it holds that*

$$\infty > \mathrm{TV}(u) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} |\xi_{E_i} - \xi_{E_j}| \mathcal{H}^{d-1}(\partial E_i \cap \partial E_j)$$

where ∂E_i denotes the boundary of E_i.

Proof. See Appendix of [73] for the proof. The so-called reduced boundary therein can be replaced in the statement by the boundary as we assume a partition into polyhedra and thus obtain that the $d-1$-dimensional Hausdorff measure is the same for both boundary definitions. □

Chapter 3

Origin of the trust-region subproblems

In this chapter we introduce the motivating class of IOCPs and the trust-region approach proposed by [66] for the one-dimensional case to tackle this problem class which was later extended to the multi-dimensional case in [73].

Let $\alpha > 0$ and $\Omega \subset \mathbb{R}^2$ be an open (axis-aligned) rectangular domain. The IOCP reads

$$\text{(IOCP)} \qquad \begin{aligned} \min_{u \in L^2(\Omega)} \quad & J(u) \coloneqq F(u) + \alpha \mathrm{TV}(u) \\ \text{s.t.} \quad & u(x) \in \Xi \coloneqq \{\xi_1, \dots, \xi_m\} \subset \mathbb{Z} \text{ for almost all (a.a.) } x \in \Omega. \end{aligned}$$

The function $F : L^2(\Omega) \to \mathbb{R}$ is lower semicontinuous, bounded from below and we assume additional regularity which we will present in Assumption 3.1 when discussing the convergence of the algorithm. The term $\mathrm{TV} : L^1(\Omega) \to [0, \infty]$ denotes the total variation seminorm. The set Ξ contains all possible control values $\xi_1, \dots, \xi_m$ and thus enforces integrality constraints on the control function. In this work we assume Ξ to be contiguous, meaning that if Ξ contains two integer values, it also contains all integer values in between. Many results hold without this restriction and as such are formulated as well as proven for this more general case.

In the following sections we will introduce the trust-region algorithm, briefly state the relevant convergence results from [73] and derive the discretized subproblems which are the main focus of this work.

3.1 The trust-region algorithm

The trust-region algorithm is detailed in Algorithm 1 and is named SLIP in [66]. The algorithm consists of a nested loop. The trust-region subproblem

$$
\begin{aligned}
pr(u,d) \coloneqq - \min_{d \in L^2(\Omega)} \; & (\nabla F(u), d)_{L^2(\Omega)} + \alpha \mathrm{TV}(u+d) - \alpha \mathrm{TV}(u) \\
\text{(TR)} \qquad \text{s.t.} \quad & u(x) + d(x) \in \Xi \text{ for a.a. } x \in \Omega, \\
& \|d\|_{L^1(\Omega)} \leq \Delta
\end{aligned}
$$

is solved in each inner loop to obtain a step inside of the trust region. We obtain this trust-region subproblem from a partial linearization of the objective around u, where only the F is linearized while the exact TV term is used. The predicted reduction, denoted by $pr(u,d)$, is given as the negative objective of (TR) and is always non-negative since $d \equiv 0$ is always feasible for (TR) with an objective function value of zero. If the predicted reduction is zero, the algorithm terminates. If the predicted reduction is positive and exceeds a certain fraction of the actual reduction $ared = J(u) - J(u+d)$, the step is accepted, which results in an updated control and a reset of the trust-region radius in the outer loop before a new inner loop is started. In case that the step is rejected, the trust-region radius is reduced and the inner loop starts anew.

Algorithm 1 Sketch of the trust-region algorithm from [66] (SLIP)

Input: feasible initial control $u^0 \in L^1(\Omega)$ for (IOCP) (that is $u^0(s) \in \Xi$ for a.a. $x \in \Omega$), reset trust-region radius $\Delta^0 > 0$, acceptance ratio $\theta \in (0,1)$

for $n = 1, \ldots$ **do**
  $k \leftarrow 0$, $\Delta^{n,0} \leftarrow \Delta^0$
  repeat
    $d^{n,k} \leftarrow$ minimizer of $\mathrm{TR}(u^{n-1}, \Delta^{n,k})$ ▷ Compute step.
    if $pr(u^{n-1}, d^{n,k}) = 0$ **then** ▷ The predicted reduction is zero.
      Terminate with solution u^{n-1}.
    else if $J(u^{n-1}) - J(u^{n-1} + d^{n,k}) < \theta pr(u^{n-1}, d^{n,k})$ **then** ▷ Reject step.
      $\Delta^{n,k+1} \leftarrow \Delta^{n,k}/2$, $k \leftarrow k+1$
    else ▷ Accept step.
      $u^n \leftarrow u^{n-1} + d^{n,k}$, $k \leftarrow k+1$
    end if
  until $J(u^{n-1}) - J(u^{n-1} + d^{n,k}) \geq \theta pr(u^{n-1}, d^{n,k})$
end for

3.2 Relevant convergence results

The major convergence results we want to briefly highlight in this section stem from the paper [73] which extends the previous convergence results restricted to the one-dimensional case from [66] to the multidimensional case. In [6] a patch-based variation of the trust-region algorithm is presented and the convergence results mimic the ones in [73]. The convergence results require additional regularity of the function F stated in the following assumption. We note that the results in [6] use a slightly different set of assumptions compared to [73] which also work for the results and are close to the assumptions found in works like [72]. For sake of completeness we state all of the assumptions and denote by A the ones stated in [73] and by B the ones stated in [6]. For this work we assume the setting B.

Assumption 3.1 (Combination of Assump. 4.3 in [73] and Assump. 4.1 in [6])**.**

1. Let $F : L^1(\Omega) \to \mathbb{R}$ be continuously Fréchet differentiable. [A,B]
2. Let $\nabla F : L^2(\Omega) \to L^2(\Omega)$ be continuously Fréchet differentiable. [A]
3. For some $C > 0$ and all $\xi \in L^2(\Omega)$, let the bilinear form induced by the Hessian $\nabla^2 F(\xi) : L^2(\Omega) \times L^2(\Omega) \to \mathbb{R}$ satisfy

   $$|\nabla^2 F(\xi)(u, w)| \leq C\|u\|_{L^1(\Omega)}\|w\|_{L^1(\Omega)} \tag{3.1}$$

   for all u, $w \in L^2(\Omega)$. [A]
4. Let $\nabla F : L^1(\Omega) \to L^\infty(\Omega)$ be Lipschitz continuous on the feasible set [B], that is

   $$\infty > L_{\nabla F} \coloneqq \sup \left\{ \frac{||\nabla F(u_1) - \nabla F(u_2)||_{L^\infty(\Omega)}}{||u_1 - u_2||_{L^1(\Omega)}} \middle| u_1, u_2 \in \operatorname{conv} \Xi \text{ for a.a. } x \in \Omega \right\}.$$

5. For all feasible $u \in \mathrm{BV}_\Xi(\Omega)$ for (IOCP) let $\nabla F(u) \in C(\bar{\Omega})$. [A,B]

In order to meaningfully discuss convergence of the algorithm, we must first establish a suitable notion of optimality. With regards to this, the authors of [66] and [73] introduce the concepts of r-optimality and stationarity.

Definition 3.2 (Definition 4.1 in [73])**.** Let $u \in \mathrm{BV}_\Xi(\Omega)$ be feasible for (IOCP). Then u is r**-optimal** for some $r > 0$ if

$$F(u) + \alpha \mathrm{TV}(u) \leq F(\tilde{u}) + \alpha \mathrm{TV}(\tilde{u})$$

holds for all $\tilde{u} \in \mathrm{BV}_\Xi(\Omega)$ that are feasible for (IOCP) and satisfy $\|u - \tilde{u}\|_{L^1(\Omega)} \leq r$.

Definition 3.3 (Definition 3.35 and Definition 4.16 in [2])**.** Let $\Omega \subset \mathbb{R}^d$ be an open set and $E \subset \mathbb{R}^d$ be a Lebesgue-measurable set. The **perimeter** $P(E, \Omega)$ of E in Ω is given by

$$P(E,\Omega) \coloneqq \sup\left\{ \int_E \operatorname{div} \phi(x) dx \;\middle|\; \phi \in C_c^1(\Omega;\mathbb{R}), ||\phi||_{L^\infty(\Omega;\mathbb{R}^d)} \leq 1 \right\}.$$

A partition $\{E_i\}_{i \in I \subset \mathbb{N}}$ of Ω is called a **Caccioppoli partition** if $\sum_i P(E_i, \Omega) < \infty$.

Definition 3.4 (Definition 4.4 in [73])**.** Let F satisfy Assumption 3.1. Let $\{E_1, \ldots, E_M\}$ be Caccioppoli partition of Ω and $u = \sum_{i=1}^m \xi_i \chi_{E_i} \in \mathrm{BV}_\Xi(\Omega)$. Then we say that v is **stationary** if the identity

$$\begin{aligned} &\sum_{i=1}^m \xi_i \int_{\partial^* E_i \cap \Omega} (-\nabla F(u))(x)(\psi(x) \cdot n_{E_i}(x)) \,\mathrm{d}\mathcal{H}^{d-1}(x) = \\ &\qquad \alpha \sum_{i=1}^m \sum_{j=i+1}^m |\xi_i - \xi_j| \int_{\partial^* E_i \cap \partial^* E_j} \operatorname{div}_{E_i} \psi(x) \,\mathrm{d}\mathcal{H}^{d-1}(x). \end{aligned} \tag{3.2}$$

holds for all $\psi \in C_c^\infty(\Omega;\mathbb{R}^d)$ where $\partial^* E$ denotes the reduced boundary of E, see Definition 3.54 in [2].

Remark 3.5. We do not formally introduce the concept of reduced boundary here as we do not require it for the remaining work and is a topic outside the scope of this work which would require the introduction of many concepts from measure theory. The reduced boundary $\partial^* E$ can be viewed as the set of points for which a notion of a inner normal vector to the set E exists.

It is straightforward to see that r-optimality is a necessary condition for global optimality, but not a sufficient condition. In Theorem 4.6 of [73] it is proven that given Assumption 3.1 r-optimality implies stationarity. Thus, stationarity is also a necessary optimality condition and proves to be a fitting concept to describe the behaviour of the iterates produced by Algorithm 1.

Theorem 3.6 (Theorem 6.4 in [73] and Theorem 5.8 in [6] with one patch)**.** *Let F be bounded from below. Let Assumption 3.1 hold. Let the iterates u^n for $n \in \mathbb{N}$ be produced by Algorithm 1. Then all iterates are feasible for* (IOCP) *and the objective values $J(u^n)$ for $n \in \mathbb{N}$ are monotonically decreasing. Moreover, one of the following mutually exclusive outcomes holds:*

1. *The number of iterates u^n is finite. The final element u^M solves the trust-region subproblem* TR *with input $u^M, \nabla F(u^M)$ and Δ for some $\Delta > 0$ and is stationary.*
2. *The number of iterates u^n is finite and the inner loop does not terminate for the final element u^M, which is stationary.*

3. *The sequence* $\{u^n\}_{n\in\mathbb{N}}$ *has a weak-* accumulation point in* $\mathrm{BV}(\Omega)$ *and every weak-* accumulation point of* $\{u^n\}_{n\in\mathbb{N}}$ *is feasible. If* u *is a weak-* accumulation point of* $\{u^n\}_{n\in\mathbb{N}}$ *(that satisfies* $\nabla F(u) \in C(\Omega)$*), then it is stationary.*

3.3 The discretized subproblem

In order to solve the subproblems (TR) we need to discretize the problems. The focus of this dissertation are the following problems we obtain from a uniform discretization of the two-dimensional domain into $N \times M$ square cells $T_{i,j}$ of equal size with $i \in [N], j \in [M]$, where $N, M \in \mathbb{N}$. We use a piecewise constant ansatz for u and d with the slight abuse of notation $u_{i,j} = \frac{1}{|T_{i,j}|}\int_{T_{i,j}} u(x)\mathrm{d}x$ and $d_{i,j} = \frac{1}{|T_{i,j}|}\int_{T_{i,j}} d(x)\mathrm{d}x$. After discretization the trust-region constraint becomes $\sum_{i=1}^{N}\sum_{j=1}^{M}\lambda_{i,j}|d_{i,j}| \leq \Delta$ where $\lambda_{i,j}$ is the Lebesgue measure of the cell $T_{i,j}$. Because all cells have the same size and thus the same Lebesgue measure, we can assume without loss of generality that $\sum_{i=1}^{N}\sum_{j=1}^{M}|d_{i,j}| \leq \Delta$ (otherwise we scale the inequality by one over the value of the Lebesgue measure and obtain a new right hand side which we use as the trust-region radius instead). The discretized version of the first term in the objective $(\nabla F(u), d)_{L^2(\Omega)}$ turns into $\sum_{i=1}^{N}\sum_{j=1}^{M} d_{i,j}\int_{T_{i,j}}\nabla F(u)(x)\mathrm{d}x = \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}d_{i,j}$ where $c_{i,j} \coloneqq \int_{T_{i,j}}\nabla F(u)(x)dx$. From Lemma 2.17 we know that

$$\begin{aligned}\mathrm{TV}(u+d) = &\sum_{i=1}^{N-1}\sum_{j=1}^{M}|u_{i+1,j} + d_{i+1,j} - u_{i,j} - d_{i,j}|\mathcal{H}^{d-1}(\partial T_{i,j} \cap \partial T_{i+1,j}) \\ &+ \sum_{i=1}^{N}\sum_{j=1}^{M-1}|u_{i,j+1} + d_{i,j+1} - u_{i,j} - d_{i,j}|\mathcal{H}^{d-1}(\partial T_{i,j} \cap \partial T_{i,j+1}).\end{aligned}$$

The term $\mathcal{H}^{d-1}(\partial T_{i,j} \cap \partial T_{i+1,j})$ can be interpreted as the length of the shared side of the square cells $T_{i,j}$ and $T_{i+1,j}$. Because the Hausdorff measure of the intersection of the boundaries is the length of the intersection for square cells in the two-dimensional case, see Theorem 2.2 in [40], and all these lengths are equal we obtain

$$\alpha\sum_{i=1}^{N-1}\sum_{j=1}^{M}|u_{i+1,j} + d_{i+1,j} - u_{i,j} - d_{i,j}| + \alpha\sum_{i=1}^{N}\sum_{j=1}^{M-1}|u_{i,j+1} + d_{i,j+1} - u_{i,j} - d_{i,j}|$$

as the discretized version of the second objective term where we assumed that the obtained weights with the same value are all equal to 1 (otherwise we scale the final cost function accordingly). The third objective term is constant and can thus be dropped for the optimization.

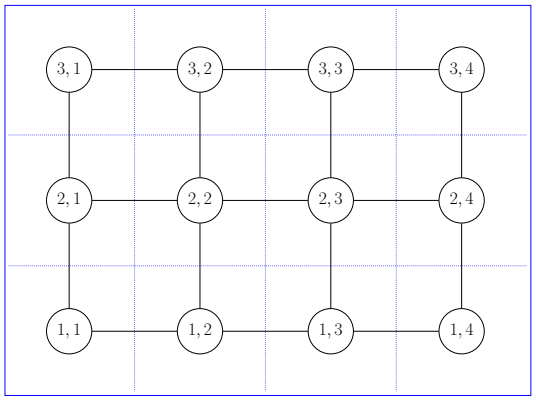

Figure 3.1: The rectangular domain is discretized into 12 square cells of equal size and induces an underlying grid with $N = 3$ and $M = 4$. For each node there exists a corresponding entry in the control $u + d$. For each edge there exists a corresponding absolute value term (horizontal: $\beta_{i,j}$, vertical: $\gamma_{i,j}$ in Section 4.1) modeling the contribution of the control jump between neighbouring cells to the total variation.

It follows that the resulting trust-region subproblems have the form

$$\begin{aligned} \min_{d \in \mathbb{Z}^{N \times M}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{M} c_{i,j} d_{i,j} + \alpha \sum_{i=1}^{N-1} \sum_{j=1}^{M} |u_{i+1,j} + d_{i+1,j} - u_{i,j} - d_{i,j}| \\ & + \alpha \sum_{i=1}^{N} \sum_{j=1}^{M-1} |u_{i,j+1} + d_{i,j+1} - u_{i,j} - d_{i,j}| \\ \text{s.t.} \quad & u_{i,j} + d_{i,j} \in \Xi \text{ for all } i \in [N], j \in [M] \quad \text{and} \quad \sum_{i=1}^{N} \sum_{j=1}^{M} |d_{i,j}| \leq \Delta \end{aligned} \tag{TR-IP}$$

with $c \in \mathbb{R}^{N \times M}$, $u \in \Xi^{N \times M}$, $\alpha > 0$ and we choose $\Delta \in \mathbb{N}$. In the case of a one-dimensional domain the above formulation holds with $M = 1$ or $N = 1$. Otherwise, we obtain a two-dimensional problem where the underlying structure can be interpreted as an $N \times M$ grid as depicted in Figure 3.1. The discretized trust-region constraint $\sum_{i=1}^{N} \sum_{j=1}^{M} |d_{i,j}| \leq \Delta$ is in the following referred to as the capacity constraint. We have dropped a constant term corresponding to $\mathrm{TV}(u)$ in (TR) from the objective as this does not affect the optimal solution. By introducing auxiliary slack variables we can model the absolute value terms in the cost function and in the trust-region constraint as linear inequalities. This leads to a reformulation as a (mixed-) integer linear program which is the focus of this work and introduced in the next chapter. Before doing so, we give a remark on the discretization that in particular points out its limitations due to the underlying geometry of our discretization of Ω into square cells.

Remark 3.7. Using a uniform discretization of the problem (TR) with piecewise constant functions recovers the correct total variation when bringing the mesh size to 0 in the one-dimensional case. In the two-dimensional case an approximation obtained this way does not necessarily recover the correct total variation in the limit. Counterexamples can be found in the literature for different discretizations. In Example 4.1 in [8] a triangulation of a rectangular domain Ω into squares and triangles as well as an $u \in L^1(\Omega)$ are given. It is shown that there is no sequence $(u_n)_{n\in\mathbb{N}} \subset L^1(\Omega)$ constant on each part of the triangulation which both converges in $L^1(\Omega)$ to the given limit $u \in L^1(\Omega)$ and recovers the correct value for the total variation. In Example 3.6 in [27] the inapproxibility was also shown for a rectangular domain discretized into uniform square cells. The study of the presented discretized problems is important.

On one hand, one may instead obtain convergence to an anisotropic version of the total variation. In [31] it was shown that for the presented problem class in Section 5 of [31] any sequence of minimizers (u_h) for the discretized problems obtained on a triangulation $\mathcal{T}_h$ of the domain with piecewise constant functions has a subsequence which converges strongly to a solution to the anisotropic version of the original problem $\bar{u}$ in L^p and the total variation of the minimizers converges to this anisotropic version of the total variation of $\bar{u}$. In detail we can generalize the total variation by

$$\mathrm{TV}_\psi(u) \coloneqq \sup\left\{ \int_\Omega u(x) \operatorname{div} \phi(x) dx \ : \ \phi \in C_c^1(\Omega;\mathbb{R}^d), \psi^*(\phi(x)) \leq 1 \text{ for all } x \in \Omega \right\}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is convex and positively one-homogeneous function such that $\phi(x) > 0$ for $x \neq 0$.

A variety of papers in imaging concern themselves with the study of anisotropic versions of the total variation for the so-called ROF model, see [86]. Examples include but are not limited to [39], [65] and [9]. Thus, a variation of the trust-region approach with regards to an anisotropic functional instead of the total variation may be desirable.

On the other hand, the problem (TR-IP) is key for the approach proposed in [92] to recover the correct total variation in the limit. The authors of [92] introduce a discretization approach with two different superlinearly coupled meshes. This induces an averaging effect which allows for the recovery of the total variation of an integer-valued function. The problem (TR-IP) is the initial problem needed to be solved for this solution approach when combined with the trust-region approach and is amended by additional linear constraints in the later iterations based on which constraint is most-violated. Thus, one needs to repeatedly solve integer programs increasing in size. To reduce the computational demand one needs to take into account the information like branching decisions gained by previous integer programs. Alternatively, one could also solve the (TR-IP) problem and add the most-violated constraint as a lazy constraints whenever a new feasible point with an improved ob-

jective value is found, which means that only one mixed-integer program has to be solved. We see that it is necessary, but maybe not suffient, for the approach in [92] to reduce the computational demand for the problem (TR-IP).

Chapter 4

The integer programming formulation and relaxations

As mentioned in the previous chapter we can reformulate our problem (TR-IP) as a (mixed-)integer program by introducing auxiliary variables and additional linear constraints. This (mixed-)integer program and the corresponding linear programming relaxation are introduced in Section 4.1. In Section 4.2 and Section 4.3 we introduce two additional relaxations. The first relaxation is obtained in Section 4.2 from a relaxation of the trust-region constraint and will be referred to as the Lagrangian relaxation throughout the remainder. Afterwards in Section 4.3, we use a dual decomposition approach to split a variation of the integer program into two efficiently solvable problems to obtain an additional relaxation, see [10] for an overview of Lagrangian relaxations and decomposition ideas. The sections are derived from [74].

4.1 The integer programming formulation

We introduce the auxiliary variables $\beta \in \mathbb{R}_{\geq 0}^{(N-1)\times M}$ and $\gamma \in \mathbb{R}_{\geq 0}^{N\times(M-1)}$ to model the absolute value terms which correspond to the total variation as linear constraints and the variables $\delta \in \mathbb{R}_{\geq 0}^{N\times M}$ to model the capacity constraint with one linear inequality. In the context of the underlying grid structure depicted in Figure 3.1, the variables β and γ correspond to the edges between nodes and model the row and column jumps respectively. The capacity consumptions are given by the entries of δ. We obtain the problem

$$\text{(IP)} \qquad \begin{aligned} \min_{d,\delta,\beta,\gamma} \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}d_{i,j} + \alpha\left(\sum_{i=1}^{N-1}\sum_{j=1}^{M}\beta_{i,j} + \sum_{i=1}^{N}\sum_{j=1}^{M-1}\gamma_{i,j}\right) \\ \text{s.t.} \quad & (d,\delta,\beta,\gamma) \in P_\Delta \text{ and } d \in \mathbb{Z}^{N\times M}, \end{aligned}$$

where $P_\Delta = \{(d, \delta, \beta, \gamma) \in P \mid \sum_{i=1}^{N} \sum_{j=1}^{M} \delta_{i,j} \leq \Delta\}$ is the polyhedron obtained from the intersection of the capacity constraint and P defined by

$$
\begin{aligned}
&(d, \delta, \beta, \gamma) \in P \\
&\Leftrightarrow \begin{cases}
\min \Xi \leq u_{i,j} + d_{i,j} \leq \max \Xi & \text{for all } i \in [N], j \in [M], \\
-\beta_{i,j} \leq u_{i+1,j} + d_{i+1,j} - u_{i,j} - d_{i,j} \leq \beta_{i,j} & \text{for all } i \in [N-1], j \in [M], \\
-\gamma_{i,j} \leq u_{i,j+1} + d_{i,j+1} - u_{i,j} - d_{i,j} \leq \gamma_{i,j} & \text{for all } i \in [N], j \in [M-1], \\
-\delta_{i,j} \leq d_{i,j} \leq \delta_{i,j} & \text{for all } i \in [N], j \in [M],
\end{cases}
\end{aligned}
$$

where we recall $u \in \Xi^{N \times M}$ so that there is always a feasible tuple with $d \equiv 0$ and $\delta \equiv 0$. The corresponding linear programming relaxation reads

$$
\text{(LP)} \qquad \begin{aligned}
\min_{d,\delta,\beta,\gamma} \quad & \sum_{i=1}^{N} \sum_{j=1}^{M} c_{i,j} d_{i,j} + \alpha \left(\sum_{i=1}^{N-1} \sum_{j=1}^{M} \beta_{i,j} + \sum_{i=1}^{N} \sum_{j=1}^{M-1} \gamma_{i,j} \right) \\
\text{s.t.} \quad & (d, \delta, \beta, \gamma) \in P_\Delta.
\end{aligned}
$$

Remark 4.1 (Remark 1 in Section 3.1 in [74])**.** A feasible point $(d, \delta, \beta, \gamma)$ can only be optimal for (IP) if $\beta_{i,j} = |u_{i+1,j} + d_{i+1,j} - u_{i,j} - d_{i,j}|$ and $\gamma_{i,j} = |u_{i,j+1} + d_{i,j+1} - u_{i,j} - d_{i,j}|$ because otherwise we could reduce the objective value by setting the values of β and γ to those absolute values. Furthermore, if $\delta_{i,j} > |d_{i,j}|$ we can always choose the minimal $\delta_{i,j} = |d_{i,j}|$ and remain feasible. Thus, we can construct the corresponding δ, β and γ from a given d.

Consequently, if we say that d^* is optimal or feasible, we mean that the point $(d^*, \delta^*, \beta^*, \gamma^*)$ is optimal or feasible when δ^*, β^* and γ^* are determined as above.

Remark 4.2 (Remark 4 in [96])**.** We note that the input $\Delta \in \mathbb{N}$ can be restricted to

$$
\Delta_{\max} \coloneqq (\max \Xi - \min \Xi) NM \tag{4.1}
$$

because the constraint $\sum_{i=1}^{N} \sum_{j=1}^{M} \delta_i \leq \Delta$ may be dropped in (IP) if $\Delta \geq \Delta_{\max}$.

Remark 4.3. In the case of a binary control set, meaning $\Xi = \{0, 1\}$, we do not need the variables δ to model the problem. Instead, we can replace each $\delta_{i,j}$ by $d_{i,j}$ if $u_{i,j} = 0$ and by $-d_{i,j}$ if $u_{i,j} = 1$. This reduction in variables can also be used for arbitrary contiguous Ξ for entries in δ for which the corresponding entry in u is either the smallest or largest value in Ξ, respectively. We also note that we can always assume $\alpha = 1$ as we could just scale the objective function by $\frac{1}{\alpha}$ without changing the feasible or optimal points.

4.2 The Lagrangian relaxation

In this section we discuss the Lagrangian relaxation we obtain from relaxing the capacity constraint and moving it to the objective as a penalty term. As noted previously, we refer to this relaxation as the *Lagrangian relaxation* for the remainder of this work. We will show that this relaxation is polynomially solvable and provides a conditional p-approximation. We end up with the problem formulation

(LR-Δ)

$$\max_{\mu\geq 0}\min_{d,\delta,\beta,\gamma} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}d_{i,j} + \alpha\left(\sum_{i=1}^{N-1}\sum_{j=1}^{M}\beta_{i,j} + \sum_{i=1}^{N}\sum_{j=1}^{M-1}\gamma_{i,j}\right) + \mu\left(\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{i,j} - \Delta\right)$$
$$\text{s.t.} \quad (d,\delta,\beta,\gamma)\in P \text{ and } d\in\mathbb{Z}^{N\times M}.$$

The parameter $\mu \geq 0$ penalizes the capacity consumption and we can ensure that an optimal solution adheres to the capacity constraint by choosing μ large enough. We will see in Section 5.2 that for a fixed μ the inner minimization problem can be solved in polynomial time and the optimal μ can be determined with a binary search, see Section 8.1.1 for the one-dimensional case which extends to two dimensions. An optimal solution to this relaxation provides, based on the used capacity, a p-approximation for the problem

(OG-TR-IP)

$$\begin{aligned}\min_{d\in\mathbb{Z}^{N\times M}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}d_{i,j} + \alpha\sum_{i=1}^{N-1}\sum_{j=1}^{M}|u_{i+1,j}+d_{i+1,j}-u_{i,j}-d_{i,j}| \\ & + \alpha\sum_{i=1}^{N}\sum_{j=1}^{M-1}|u_{i,j+1}+d_{i,j+1}-u_{i,j}-d_{i,j}| \\ & - \alpha\sum_{i=1}^{N-1}\sum_{j=1}^{M}|u_{i+1,j}-u_{i,j}| - \alpha\sum_{i=1}^{N}\sum_{j=1}^{M-1}|u_{i,j+1}-u_{i,j}| \\ \text{s.t.} \quad & u_{i,j}+d_{i,j}\in\Xi \text{ for all } i\in[N], j\in[M] \\ & \sum_{i=1}^{N}\sum_{j=1}^{M}|d_{i,j}|\leq\Delta,\end{aligned}$$

which is the problem (TR-IP) before dropping the last two constant terms corresponding to $-\alpha\mathrm{TV}(u)$ in (TR) from the objective. A p-approximation guarantee for this problem, which is the negative predicted reduction in the trust-region algorithm, allows to use feasible points satisfying the p-approximation in similar ways as Cauchy points instead of optimal points in trust-region algorithms while retaining

the convergence properties. For the sake of clarity we now define

$$\omega(d) \coloneqq \alpha \sum_{i=1}^{N-1} \sum_{j=1}^{M} |u_{i+1,j} + d_{i+1,j} - u_{i,j} - d_{i,j}| + \alpha \sum_{i=1}^{N} \sum_{j=1}^{M-1} |u_{i,j+1} + d_{i,j+1} - u_{i,j} - d_{i,j}|$$

and

$$\bar{C} \coloneqq \alpha \sum_{i=1}^{N-1} \sum_{j=1}^{M} |u_{i+1,j} - u_{i,j}| + \alpha \sum_{i=1}^{N} \sum_{j=1}^{M-1} |u_{i,j+1} - u_{i,j}|.$$

Theorem 4.4 (Conditional p-approximation, Theorem 2 in Section 3.3 in [74])**.** *Let* $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma}, \bar{\mu})$ *be optimal for* (LR-Δ) *and let* d^* *be optimal for* (OG-TR-IP)*. If* $\Delta \geq \sum_{i=1}^{N} \sum_{j=1}^{M} \bar{\delta}_{i,j} \geq p\Delta$ *for* $p \in (0, 1]$ *then it holds that*

$$c^T \bar{d} + \omega(\bar{d}) - \bar{C} \leq p(c^T d^* + \omega(d^*) - \bar{C}) \leq 0.$$

Proof. Let $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma}, \bar{\mu})$ be optimal for (LR-Δ) and we note that $p \in (0, 1]$ gives that $\bar{d}$ is also feasible for (OG-TR-IP). We first prove the case $p = 1$. The inequality

$$c^T \bar{d} + w(\bar{d}) - \bar{C} + \bar{\mu}(\sum_{i=1}^{N} \sum_{j=1}^{M} \bar{\delta}_{i,j} - \Delta) \leq c^T d + w(d) - \bar{C}$$

holds for every d feasible for (OG-TR-IP). If $\sum_{i=1}^{N} \sum_{j=1}^{M} \bar{\delta}_{i,j} = \Delta$, then the objective values of (OG-TR-IP) and (LR-Δ) coincide.

We now turn to the case $p \in (0, 1)$. We prove this result by way of contradiction and assume

$$c^T \bar{d} + \omega(\bar{d}) - \bar{C} > p(c^T d^* + \omega(d^*) - \bar{C}), \tag{4.2}$$

where we note that both the left and right hand side of (4.2) are non-positive because $d = 0$ is feasible for both optimization problems. Because $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma}, \bar{\mu})$ is optimal for (LR-Δ), it holds that

$$c^T \bar{d} + \omega(\bar{d}) - \bar{C} + \bar{\mu} \sum_{i=1}^{N} \sum_{j=1}^{M} \bar{\delta}_{i,j} \leq c^T d^* + \omega(d^*) - \bar{C} + \bar{\mu} \sum_{i=1}^{N} \sum_{j=1}^{M} \delta^*_{i,j}. \tag{4.3}$$

Using the two inequalities (4.2) and (4.3) as well as $\sum_{i=1}^{N} \sum_{j=1}^{M} \bar{\delta}_{i,j} \geq p\Delta$ and $\sum_{i=1}^{N} \sum_{j=1}^{M} \delta^*_{i,j} \leq \Delta$, we obtain that

$$(1 - p)(c^T d^* + \omega(d^*) - \bar{C}) > \bar{\mu} \sum_{i=1}^{N} \sum_{j=1}^{M} (\bar{\delta}_{i,j} - \delta^*_{i,j}) \geq -(1 - p)\bar{\mu}\Delta.$$

After multiplication with $\frac{p}{1-p} > 0$ we obtain that

$$p(c^T d^* + \omega(d^*) - \bar{C}) > -\bar{\mu} p \Delta. \tag{4.4}$$

The point $\tilde{d} \equiv 0$ is feasible for (LR-Δ) with an objective value of $\bar{C} - \bar{\mu}\Delta$ and thus

$$c^T \bar{d} + \omega(\bar{d}) - \bar{C} + \bar{\mu}(\sum_{i=1}^{N} \sum_{j=1}^{M} \bar{\delta}_{i,j} - \Delta) \leq -\bar{\mu}\Delta$$

holds. From $\sum_{i=1}^{N} \sum_{j=1}^{M} \bar{\delta}_{i,j} - \Delta \geq (p-1)\Delta$ it follows that

$$c^T \bar{d} + \omega(\bar{d}) - \bar{C} \leq -\bar{\mu} p \Delta. \tag{4.5}$$

Combining (4.4) and (4.5) we obtain the contradiction

$$c^T \bar{d} + \omega(\bar{d}) - \bar{C} < p(c^T d^* + \omega(d^*) - \bar{C}).$$

□

The above proof does not use specific properties of the objective function or the constraint and can thus be generalized as done in Proposition 12 in [104] which references this proof.

Remark 4.5 (Remark 2 in Section 3.3 in [74])**.** Theorem 4.4 provides a p-approximation if at least a fraction p of the capacity in the form of the trust-region radius is used by the optimal solution to (LR-Δ). Intuitively, this holds when the LP-relaxation is already integer-valued in many entries of d and thus close to the solution to (LR-Δ). Note that they always coincide in the integer entries of the LP-relaxation, which follows from the proof of Theorem 7.9. Since (LR-Δ) can generally be solved much faster than (IP) or (OG-TR-IP) respectively, this could in theory be used to accept steps faster within the superordinate trust-region algorithm. In practice, this approach did not produce significant improvements in our numerical experiments. A possible explanation for this is that the quality of the approximation is dependent on the number of integer-valued entries in the linear programming solution which is a hint for a smaller gap between the lower bound and a solution to (IP) and more easily solvable mixed-integer program.

The following corollary restates the case $p = 1$ and was proven in Proposition 14 in [96] for the one-dimensional case, now extended to the two-dimensional case.

Corollary 4.6 (Corollary 2 in Section 3.3 in [74])**.** *Let* $(d^*, \delta^*, \beta^*, \gamma^*, \mu^*)$ *be optimal for* (LR-Δ) *and* $\sum_{i=1}^{N} \sum_{j=1}^{M} \delta_{i,j}^* = \Delta$ *hold. Then* $(d^*, \delta^*, \beta^*, \gamma^*)$ *is optimal for* (IP)*.*

In general, it holds for integer programming that the lower bound obtained from a solution to a Lagrangian relaxation is at least as good a bound as the bound obtained

by a solution to the linear programming relaxation, see for example [46]. In the case that the solution to a Lagrangian relaxation without the integrality conditions is still integer-valued the bounds coincide, see Theorem 2 in [46]. We will show in Chapter 7 that this is true for the given Lagrangian relaxation.

4.3 The dual decomposition

In (IP) the terms modeling the total variation are split into sums over the auxiliary variables $\beta \in \mathbb{N}^{(N-1)\times M}$ and $\gamma \in \mathbb{N}^{N\times(M-1)}$. The corresponding linear inequalities are coupled through d. To obtain a new relaxation, we rewrite (IP) such that we split each entry in d into two copies d^c and d^r where d^c is used for the inequalities containing β and d^r for the inequalities containing γ. To ensure equivalence to (IP), we add coupling equalities to enforce that both copies are equal. Dropping the coupling constraints will allow us to split the problem and solve the two resulting problems in polynomial time. Furthermore, this will provide a potentially tighter lower bound than the one obtained from (LP). For the underlying grid illustrated in Figure 3.1 this means that d^r considers only the rows while d^c considers only the columns. We obtain:

(IP-DD)

$$\begin{aligned}\min_{d^r,d^c,\delta^r,\delta^c,\beta,\gamma} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}d^r_{i,j} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}d^c_{i,j} + \alpha\left(\sum_{i=1}^{N-1}\sum_{j=1}^{M}\beta_{i,j} + \sum_{i=1}^{N}\sum_{j=1}^{M-1}\gamma_{i,j}\right)\\ \text{s.t.} \quad & (d^r,\delta^r,\gamma)\in P_r,\ (d^c,\delta^c,\beta)\in P_c,\ d^r,\ d^c\in\mathbb{Z}^{N\times M},\ \text{and } d^c = d^r,\end{aligned}$$

where P_r is the polyhedron defined by

$$(d^r,\delta^r,\gamma)\in P_r$$
$$\Leftrightarrow \begin{cases} \min\Xi \le u_{i,j} + d^r_{i,j} \le \max\Xi & \text{for all } i\in[N], j\in[M],\\ -\gamma_{i,j} \le u_{i,j+1} + d^r_{i,j+1} - u_{i,j} - d^r_{i,j} \le \gamma_{i,j} & \text{for all } i\in[N], j\in[M-1],\\ -\delta^r_{i,j} \le d^r_{i,j} \le \delta^r_{i,j} & \text{for all } i\in[N], j\in[M],\\ \sum_{i=1}^{N}\sum_{j=1}^{M}\delta^r_{i,j} \le \Delta, \end{cases}$$

and P_c is the polyhedron defined by

$$(d^c,\delta^c,\beta)\in P_c$$
$$\Leftrightarrow \begin{cases} \min\Xi \le u_{i,j} + d^c_{i,j} \le \max\Xi & \text{for all } i\in[N], j\in[M],\\ -\beta_{i,j} \le u_{i+1,j} + d^c_{i+1,j} - u_{i,j} - d^c_{i,j} \le \beta_{i,j} & \text{for all } i\in[N-1], j\in[M],\\ -\delta^c_{i,j} \le d^c_{i,j} \le \delta^c_{i,j} & \text{for all } i\in[N], j\in[M],\\ \sum_{i=1}^{N}\sum_{j=1}^{M}\delta^c_{i,j} \le \Delta. \end{cases}$$

We briefly state that the constructed problem (IP-DD) is equivalent to (IP).

Lemma 4.7 (Lemma 1 in Online Supplement B in [74]). *Let $(d, \delta, \beta, \gamma)$ be feasible for* (IP)*. Then $(d^r, d^c, \delta^r, \delta^c, \beta, \gamma)$ with $d^r = d^c = d$, $\delta^r = \delta^c = \delta$ is feasible for* (IP-DD) *with the same objective value. If $(d^r, d^c, \delta^r, \delta^c, \beta, \gamma)$ is feasible for* (IP-DD)*, then $(d, \delta, \beta, \gamma)$ with $d = d^r$ and $\delta = \delta^r$ is feasible for* (IP) *with the same objective value.*

Proof. Let $(d, \delta, \beta, \gamma)$ be feasible for (IP). We show that $(d^r, d^c, \delta^r, \delta^c, \beta, \gamma)$ with $d^r = d^c = d$ and $\delta^r = \delta^c = \delta$ is feasible for (IP-DD). We first note that $(d^r, \delta^r, \gamma) \in P_r$, which follows from $d = d^r$, $\delta = \delta^r$, and $(d, \delta, \beta, \gamma) \in P_\Delta$. By the same argumentation $(d^c, \delta^c, \beta) \in P_c$. The remaining constraints are obviously fulfilled. The objective values coincide because $d^r = d^c = d$ so that

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j} d^r_{i,j} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j} d^c_{i,j} = \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j} d_{i,j}$$

and the remaining terms are the same. The other implication can be proven in the same way by just switching the roles of d^r and d as well as of δ^r and δ and using $d^r = d^c$. □

Corollary 4.8 (Corollary 1 in Online Supplement B in [74]). *Let $(d, \delta, \beta, \gamma)$ be a feasible point of* (LP)*. Then $(d^r, d^c, \delta^r, \delta^c, \beta, \gamma)$ with $d^r = d^c = d$ and $\delta^r = \delta^c = \delta$ is feasible for the linear programming relaxation of* (IP-DD) *with the same objective value. On the other hand, if $(d^r, d^c, \delta^r, \delta^c, \beta, \gamma)$ is feasible for the linear programming relaxation of* (IP-DD)*, then $(d, \delta, \beta, \gamma)$ with $d = d^r$ and $\delta = \delta^r$ is feasible for* (LP) *with the same objective value.*

Proof. We argue as in Lemma 4.7. We just need to drop the integrality conditions from both problems. □

Remark 4.9 (Remark 1 in Online Supplement B in [74]). The previous lemma and corollary also hold for $d = d^c$ and $\delta = \delta^c$ by symmetry of the problem.

We can now obtain a Lagrangian relaxation by moving the coupling constraint $d^r = d^c$ into the objective with a corresponding multiplier. The problem reads

(LR-DD(λ))

$$\begin{aligned}\min_{d^r, d^c, \delta^r, \delta^c, \beta, \gamma} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j} d^r_{i,j} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j} d^c_{i,j} + \alpha\left(\sum_{i=1}^{N-1}\sum_{j=1}^{M}\beta_{i,j} + \sum_{i=1}^{N}\sum_{j=1}^{M-1}\gamma_{i,j}\right) \\ & + \sum_{i=1}^{N}\sum_{j=1}^{M}\lambda_{i,j}(d^r_{i,j} - d^c_{i,j}) \\ \text{s.t.} \quad & (d^r, \delta^r, \gamma) \in P_r,\ (d^c, \delta^c, \beta) \in P_c,\ d^r,\ d^c \in \mathbb{Z}^{N\times M}\end{aligned}$$

and provides a lower bound on the objective of (IP-DD) for every $\lambda \in \mathbb{R}^{N\times M}$.

Theorem 4.10 (Theorem 1 in Online Supplement B in [74]). *Let $\lambda \in \mathbb{R}^{N\times M}$ be fixed. Then the optimal objective value of* (LR-DD(λ)) *provides a lower bound for the optimal objective value of* (IP-DD).

Proof. Let $(d^r, d^c, \delta^r, \delta^c, \beta, \gamma)$ with $d^r = d^c$ be an arbitrary, feasible point of (IP-DD). Then this point is also feasible for (LR-DD(λ)). Because $d^r = d^c$ the equality $\sum_{i=1}^{N}\sum_{j=1}^{M} \lambda_{i,j}(d^r_{i,j} - d^c_{i,j}) = 0$ holds and the objective function values of (IP-DD) and (LR-DD(λ)) coincide. Thus, the optimal objective value of (LR-DD(λ)) can not be higher than the optimal objective value of (IP-DD). □

The problem (LR-DD(λ)) can be decoupled into the integer linear programs

$$\text{(R-DD)} \qquad \begin{aligned} \min_{d^r,\delta^r,\gamma} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}(c_{i,j} + 2\lambda_{i,j})d^r_{i,j} + \alpha\sum_{i=1}^{N}\sum_{j=1}^{M-1}\gamma_{i,j} \\ \text{s.t.} \quad & (d^r, \delta^r, \gamma) \in P_r \text{ and } d^r \in \mathbb{Z}^{N\times M}, \end{aligned}$$

and

$$\text{(C-DD)} \qquad \begin{aligned} \min_{d^c,\delta^c,\beta} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}(c_{i,j} - 2\lambda_{i,j})d^c_{i,j} + \alpha\sum_{i=1}^{N-1}\sum_{j=1}^{M}\beta_{i,j} \\ \text{s.t.} \quad & (d^c, \delta^c, \beta) \in P_c \text{ and } d^c \in \mathbb{Z}^{N\times M}, \end{aligned}$$

which can be solved independently for d^r, δ^r, γ and d^c, δ^c, β in order to solve (LR-DD(λ)). Each of the resulting problems is a one-dimensional version of (TR-IP). We have already shown that the sum of the optimal values provides a lower bound. We are now interested in cases where this allows us to construct an optimal point for (IP-DD) and hence (IP).

Theorem 4.11 (Theorem 2 in Online Supplement B in [74]). *Let $(d^{r,*}, \delta^{r,*}, \gamma^*)$ be optimal for the problem* (R-DD) *and $(d^{c,*}, \delta^{c,*}, \beta^*)$ be optimal for* (C-DD). *If $d^{r,*} = d^{c,*}$ holds, then $(d^*, \delta^*, \beta^*, \gamma^*)$ with $d^* = d^{r,*}$ and $\delta^* = \delta^{r,*}$ is optimal for* (IP-DD) *and hence* (IP).

Proof. The feasibility of the constructed point follows by construction because $d^{r,*}$ and $d^{c,*}$ adhere to the capacity constraint and the controls can only take control values in Ξ. Thus, it remains to show the optimality.

Let $(d, \delta, \beta, \gamma)$ be feasible for (IP). Let $(d^{r,*}, \delta^{r,*}, \gamma^*)$ be an optimal point for (R-DD) and $(d^{c,*}, \delta^{c,*}, \beta^*)$ be an optimal point for (C-DD). Then the two inequalities

$$\sum_{i=1}^{N}\sum_{j=1}^{M}(\frac{1}{2}c_{i,j} - \lambda_{i,j})d_{i,j} + \alpha\sum_{i=1}^{N}\sum_{j=1}^{M-1}\gamma_{i,j} \geq \sum_{i=1}^{N}\sum_{j=1}^{M}(\frac{1}{2}c_{i,j} - \lambda_{i,j})d^{r,*}_{i,j} + \alpha\sum_{i=1}^{N}\sum_{j=1}^{M-1}\gamma^*_{i,j}$$

and

$$\sum_{i=1}^{N}\sum_{j=1}^{M}(\frac{1}{2}c_{i,j}+\lambda_{i,j})d_{i,j}+\alpha\sum_{i=1}^{N-1}\sum_{j=1}^{M}\beta_{i,j}\geq\sum_{i=1}^{N}\sum_{j=1}^{M}(\frac{1}{2}c_{i,j}+\lambda_{i,j})d_{i,j}^{c,*}+\alpha\sum_{i=1}^{N-1}\sum_{j=1}^{M}\beta_{i,j}^{*}$$

hold. The statement follows from adding both inequalities. □

Remark 4.12 (Remark 2 in Online Supplement B in [74])**.** The most straightforward split is into one row and one column problem. We later on provide a (pseudo-)polynomial algorithm for the one-dimensional case which can also solve these problems. Other splits of β and γ into two sets are also possible but might create subproblems which are not known to be pseudo-polynomially solvable. If the underlying graphs are trees (or forests), we will see in Section 8.2 that the resulting problems are still (pseudo-)polynomially solvable.

The bound provided by a Lagrangian relaxation with an optimal Lagrange multiplier is at least as good as the bound provided by the linear programming relaxation, see [46]. Thus, Corollary 4.8 shows that the bound from (LR-DD(λ)) for an optimal λ, which maximizes the objective of (LR-DD(λ)), is at least as good as the bound from (LP). We provide a minimal example to show that the dual decomposition relaxation can also provide a tighter bound.

Example 4.13 (Example 1 in Online Supplement B in [74])**.** Let $N = M = 2$, $\Delta = 1$, $\alpha = 1$, $\Xi = \{0, 1\}$,

$$c = \begin{pmatrix} -0.5 & -0.5 \\ -0.5 & -0.5 \end{pmatrix},\ u = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Then the unique optimal point of (R-DD) for $\lambda \equiv 0$ is $d^r \equiv 0$ and of (C-DD) for $\lambda \equiv 0$ is $d^c \equiv 0$. Thus, $d \equiv 0$ is optimal for (IP) but the linear programming relaxation solution is

$$d = \begin{pmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{pmatrix}$$

which shows that the dual decomposition relaxation is tighter in this case.

The previous example shows that it is possible in theory to obtain a tighter lower bound from this kind of relaxation. Thus, we could add a cutting plane based on this lower bound to improve the value of the linear programming relaxation. In practice, this did not prove useful in numerical experiments. The previous example works well because the trust-region radius in the example is smaller than the dimension of a column or a row. We now turn to an example for which this condition is not given and the trust-region radius is at least the dimension of a row or column.

Example 4.14. Let $N = M = 3$, $\Delta = 3$, $\alpha = 1$, $\Xi = \{0, 1\}$,

$$c = \begin{pmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{pmatrix}, \; u = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

The optimal solution for the linear programming relaxation to (IP) is given by

$$d^{LP} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}.$$

Let $\lambda \in \mathbb{R}^{3\times 3}$ be arbitrary. We see that one of the following three feasible integer points has an objective value at least as good as d^{LP} for (R-DD):

$$d^1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \; d^2 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \; d^3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

This is straightforward to see as d^{LP} is a convex combination of the three (one needs to consider the variables modelling the jumps in between entries in the same row as well, but these have the correct values if d^{LP}, d^1, d^2 and d^3 are constructed as in Remark 4.1). For (IP) d^{LP} is not a convex combination of the three as for this problem we would have to also consider the variables modeling the jumps between entries in the same columns. Thus, we can only expect a better bound if the linear programming relaxation can not be modeled by a convex combination of feasible integer points when considering the problem (R-DD) (or (C-DD)).

One might argue that this example only works because all non-integer values for $u + d$ are identical. We will see in Chapter 7 that this is a property which always holds for the solution to the linear programming relaxation. We note that this does not mean that this relaxation never provides better or significantly better bounds if the trust-region radius is significantly larger than the dimension of a row or column, but that this is an indicator for why it might not.

Chapter 5

Relevant graph-based problems

In this chapter we discuss several graph-based problems which are connected to the trust-region subproblem (TR-IP). In the first half we present the (constrained) shortest path problem and the (constrained) minimum s-t cut problem into which the problem (TR-IP) can be reformulated in the one-dimensional and two-dimensional case respectively. This first half is based on [96] which is an extension of [95] with the original graph constructions. In the second half we introduce the knapsack problem and the minimum bisection problem which are relevant for the NP-hardness proofs in Chapter 6.

5.1 (Constrained) shortest path problem

We first introduce the shortest path problem and afterwards turn to the constrained shortest path problem.

The shortest path problem (SPP): Given a graph $G = (V, E)$, a source $s \in V$, a sink $t \in V$ and a length (or weight) function $w : E \to \mathbb{R}_{\geq 0}$, (SPP) seeks for a path P from s to t such that $\sum_{e \in P} w(e)$ is minimized.

The shortest path problem can be solved in polynomial time for example by the Dijkstra algorithm, see [63].

The resource-constrained shortest path problem (RCSPP): Given a graph $G = (V, E)$, a source $s \in V$, a sink $t \in V$, a length (or weight) function $w_1 : E \to \mathbb{R}_{\geq 0}$, a resource cost (or weight) function $w_2 : E \to \mathbb{R}_{\geq 0}$ and a budget $B > 0$, (RCSPP) seeks for a path P from s to t such that $\sum_{e \in P} w_1(e)$ is minimized and $\sum_{e \in P} w_2(e) \leq B$ holds.

The problem (RCSPP) is NP-hard, see [44]. The problem is also closely related to the knapsack problem, see [54], which we introduce in Section 5.3. If the weights are integer-valued, the problem can be solved as a shortest path problem on a graph with a size dependent on the budget B. We will show that the Lagrangian relaxation (LR-Δ) can be interpreted as a shortest path problem in the one-dimensional case. The one-dimensional (TR-IP) corresponds to the (resource) constrained shortest path problem with the same graph structure but can be reformulated as a shortest path problem on a graph with a size depending on the trust-region radius Δ.

5.1.1 Reformulation of (TR-IP) as a (SPP)

We will use a more general problem for the graph construction which also applies to the problem (IP) as a special case. In the case that a non-uniform discretization is used, the one-dimensional (TR-IP) is transformed into the problem

$$\text{(1D-w)} \qquad \begin{aligned} \min_{d\in\mathbb{Z}^N} \quad & \sum_{i=1}^{N} c_i d_i + \alpha \sum_{i=1}^{N-1} |u_{i+1} + d_{i+1} - u_i - d_i| \\ \text{s.t.} \quad & u_i + d_i \in \Xi \text{ for all } i \in [N] \quad \text{and} \quad \sum_{i=1}^{N} \sigma_i |d_i| \leq \Delta \end{aligned}$$

where $\sigma \in \mathbb{R}^N_{\geq 0}$ are the weights resulting from the chosen discretization. We obtain the one-dimensional (TR-IP) by setting $\sigma \equiv 1$. In the following graph construction we assume that $\sigma \in \mathbb{N}^N$ as we need to choose $\sigma \in \mathbb{Q}^N$ to solve the problems numerically which is not an actual restriction due to $\mathbb{Q}$ being dense in $\mathbb{R}$. Thus, we can multiply all weights and the trust-region radius by the product of the denominators to obtain integer values. To obtain the equivalent (SPP) formulation of (1D-w), we introduce the parameterized family

$$\text{(1D-w}(j)\text{)} \qquad \begin{aligned} \min_{d_{j+1},\dots,d_N} \quad & \sum_{i=j+1}^{N} c_i d_i + \alpha \sum_{i=\max\{j,1\}}^{N-1} |u_{i+1} + d_{i+1} - u_i - d_i| \\ \text{s.t.} \quad & u_i + d_i \in \Xi \text{ for all } i \in \{j+1,\dots,N\}, \\ & \sum_{i=j+1}^{N} \sigma_i |d_i| \leq \Delta - \sum_{i=1}^{j} \sigma_i |d_i|. \end{aligned}$$

for all $j \in \{0,\dots,N\}$. It is immediate that (1D-w(0)) is an equivalent representation of (1D-w). For $j \in [N]$, the problem (1D-w(j)) corresponds to (1D-w) with $d_1,\dots,d_j$ being fixed. Specifically, the *resource capacity constraint* $\sum_{i=1}^{N} \sigma_i |d_i| \leq \Delta$ in (1D-w) changes to

$$\sum_{i=j+1}^{N} \sigma_i |d_i| \leq r(\Delta, d_1,\dots,d_j) \coloneqq \Delta - \sum_{i=1}^{j} \sigma_i |d_i| \in \mathbb{Z},$$

where we call $r(\Delta, d_1, \dots, d_j)$ the *remaining capacity* for the resource capacity constraint in (1D-w(j)).

Because the problem (1D-w(j)) cannot admit a feasible point if $r(\Delta, d_1, \dots, d_j) < 0$, we say that a *remaining capacity is feasible* if $r(\Delta, d_1, \dots, d_j) \geq 0$. Because $\sigma_i |d_i| \in \mathbb{N}$ for all $i \in [N]$, a feasible capacity can only assume the integer values between 0 and Δ.

The structure of the problem, specifically the shrinking feasible set of (1D-w(j)) for increasing j, allows us to construct a digraph $G = (V, A)$, where the set of nodes V is partitioned into N layers and the directed edges (in the set A) can only exist between subsequent layers, that is from layer j to $j+1$ for $j \in [N-1]$. The construction is visualized in Figure 5.1.

Nodes in G. Let $i \in [N]$, then the nodes in the layer i encode feasibility of the d_i with respect to the integrality constraint $u_i + d_i \in \Xi$ and the resource capacity constraint. Formally, a node $v \in V$ is a triplet $v = (j, \delta, \eta) \in [N] \times \{x - y \,|\, x, y \in \Xi\} \times \{0, \dots, \Delta\}$ of the layer index j, the step in the control entry δ, and the remaining capacity η. For $i \in [N]$, the layer L_i is defined as the set of triplets

$$L_i := \left\{ (i, \delta, \eta) \;\middle|\; \delta \in \Xi - u_i \text{ and } \eta \in \{\Delta - |\delta|\sigma_i, \Delta - |\delta|\sigma_i - 1, \dots, 0\} \right\}$$

and the set of nodes is $V = L_1 \dot\cup \cdots \dot\cup L_N$, where $\dot\cup$ denotes the disjoint union. To access the entries of a node $v = (j, \delta, \eta) \in V$, we define the notation

$$\ell(v) = j, \;\; \tilde{d}(v) = \delta, \;\; \text{and} \;\; \tilde{r}(v) = \eta.$$

Directed edges in G. The set $A \subset V \times V$ of directed edges is defined as

$$(v_i, v_j) \in A \quad :\Longleftrightarrow \quad \begin{cases} \ell(v_j) = \ell(v_i) + 1, \\ \text{there exists } (a, b) \in A \text{ with } b = v_i \text{ if } \ell(v_i) > 1, \text{ and} \\ \tilde{r}(v_j) = \tilde{r}(v_i) - \sigma_{\ell(v_j)} |\tilde{d}(v_j)|. \end{cases}$$

The first condition guarantees that we obtain a layered directed acyclic graph (LDAG) because edges only exist between subsequent layers, while the second and third condition ensure that the resource capacity constraint is satisfied inductively. The weight of an edge $e = (v_i, v_j)$ is given by

$$w_{(v_i, v_j)} = c_{\ell(v_j)} \tilde{d}(v_j) + \alpha |u_{\ell(v_j)} - u_{\ell(v_i)} + \tilde{d}(v_j) - \tilde{d}(v_i)|,$$

which considers the linear cost part in the first term and the jump height in the second term. Furthermore, a source s and a sink t are added to the graph. The source $s = (0, \emptyset, \Delta)$ is connected to all $v_i \in V$ in the first layer with sufficient

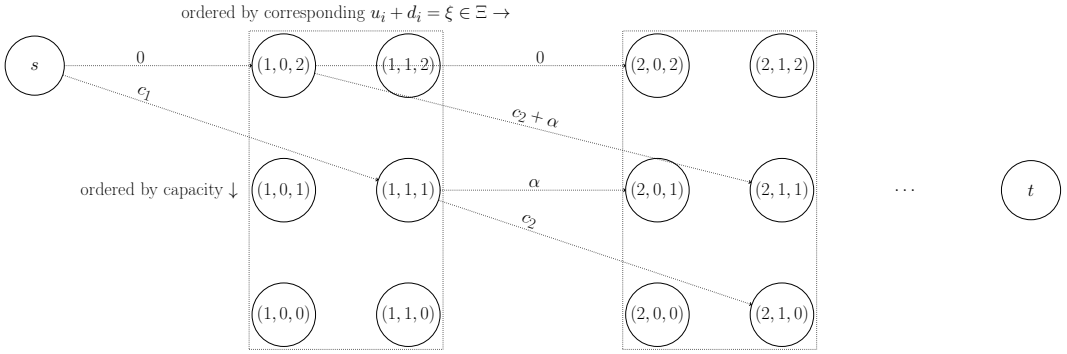

Figure 5.1: Example of the graph construction for arbitrary $c \in \mathbb{R}^N$ and $N \in \mathbb{N}_{\geq 2}$, $\Xi = \{0, 1\}$, $u_1, u_2 = 0$, $\sigma_1, \sigma_2 = 1$, $\Delta = 2$. Edges always point from left to right as they only connect subsequent layers. Because the remaining capacity is decreasing monotonously, this introduces a level structure where each level represents a remaining capacity between 0 and Δ. Edges can only point to nodes of the same or a lower level. Original in [96].

remaining capacity, that is

$$(s, v_i) \in A \quad :\Longleftrightarrow \quad \ell(v_i) = 1 \text{ and } \tilde{r}(v_i) = \Delta - |\tilde{d}(v_i)|\sigma_1.$$

Moreover, we have the weight $w_{(s,v_i)} = c_1\tilde{d}(v_i)$. The sink $t = (N+1, \emptyset, 0)$ is connected to each node $v_j \in V$ in the last layer that has an incoming edge, that is

$$(v_j, t) \in A \quad :\Longleftrightarrow \quad \text{there exists } v_i \in V \text{ such that } (v_i, v_j) \in A.$$

The weights have the value zero, that is $w_{(v_j,t)} = 0$.

Figure 5.1 shows a sketch of such a graph $G = (V, A)$. It depicts the layered structure as well as the directed edges encoding feasible choices from one layer to the next.

The construction detailed above implies immediately the following upper bounds on the cardinalities of V and A for the graph G:

$$|V| \leq N \cdot (\Delta + 1) \cdot |\Xi| + 2, \tag{5.1}$$

$$|A| \leq |\Xi|^2 \cdot (N - 1) \cdot (\Delta + 1) + |\Xi| + (\Delta + 1) \cdot |\Xi|. \tag{5.2}$$

Theorem 5.1 (Proposition 7 in [96])**.** *There is a one-to-one correspondence between solutions of* (1D-w) *and solutions of the (SPP) on G from s to t.*

Proof. See Online Supplement to [96]. □

5.1.2 Reformulation of (TR-IP) as a (RCSPP)

To obtain an (RCSPP) on a quotient graph $G_c = (V_c, A_c)$, we define an equivalence relation $\sim_G$ on the set of nodes. Let $v_i, v_j \in V$. Then we can define

$$v_i \sim_G v_j \quad :\Longleftrightarrow \quad \ell(v_i) = \ell(v_j) \quad \text{and} \quad \tilde{d}(v_i) = \tilde{d}(v_j).$$

This means that nodes in the same equivalence class only differ in their remaining capacity. We denote the equivalence class containing $v \in V$ as $[v]$. This naturally leads to a reformulation of the SPP on G as an RCSPP on the quotient graph. Equivalence classes are nodes in the new LDAG $G_c = (V_c, A_c)$ with fully connected subsequent layers. Each layer i contains a node for each feasible choice of d_i. Thus, a node is a pair of the layer i and the choice of d_i. Again, a source s and a sink t are added to the graph G_c and connected to each node of the first and last layer respectively. Since the weights of the edges in G do not depend on the remaining capacity of the incident nodes, the weights are defined exactly as for G. Additionally, we assign a resource consumption to each edge. An edge from the source s to a node v_i in the first layer has a resource consumption of $\sigma_1|\tilde{d}(v_i)|$, while an edge from a node v_i to a node v_j has a resource consumption of $\sigma_{\ell(v_j)}|\tilde{d}(v_j)|$. Edges into the sink t have a resource consumption of 0.

Theorem 5.2 (Proposition 8 in [96])**.** *There is a one-to-one correspondence between solutions of the (RCSPP) on G_c and solutions of the (SPP) on G.*

Proof. See Online Supplement to [96]. □

Remark 5.3. Because $c_i d_i < 0$ is possible, the weights of the edges may also be negative. To ensure non-negative weights a uniform offset can be added to all weights. Because all paths from s to t have the same length, the cost of all paths is increased by the same fixed amount.

It is straightforward to see that the (SPP) problem on the constructed graph corresponds to dropping the capacity constraint from (1D-w). Thus, the Lagrangian problem (LR-Δ) can be solved as an (SPP) on the graph with a different length function.

Remark 5.4. In the one-dimensional case we can consider the underlying domain as an interval in time and the discretization as the time steps (or intervals). In this case, we can introduce boundary conditions which enforce that before and after the time interval a fixed control value is given. Thus, we have to consider additional jump terms in the first and last time step. This can be encoded in the graph formulation by changing the weight on the edges from s to nodes in the first layer as well as on the edges from nodes in the last layer to t.

Another avenue of interest are dwell-time constraints which require that after the control value is changed to a given integer value it is not changed for at least a given

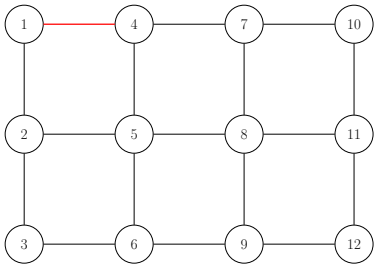

Figure 5.2: We use a shortest path approach to determine the optimal control in each node in the order 1 to 12. To guarantee optimality we need to take the control value in node 1 into account when choosing the control value in node 4 in order to accurately model the red edge between the nodes. Thus, we need to encode the last $\min\{N, M\} = 3$ control values in a graph construction which means the size grows exponentially in $\min\{N, M\}$. If all row edges are ignored, the resulting problem can be solved by a pseudo-polynomial algorithm in the same manner as the one-dimensional (TR-IP). Original in [74].

minimum time, see [21] and [105]. We can include dwell-time constraints into our graph formulation by not connecting subsequent layers but instead connecting layers with a difference between the layers corresponding to the length of the dwell times if different control values are encoded in the connected nodes.

5.2 (Constrained) minimum s-t cut problem

Unlike the one-dimensional case a shortest path approach can not be applied in two dimensions by traversing the underlying grid in a one-dimensional sequence, as either half of the terms modeling the total variation would have to be ignored or the size of the graph would have to grow exponentially to encode the necessary information of at least the previous $\min\{N, M\}$ graph layers to guarantee optimality, see Figure 5.2. Even without the capacity constraint the shortest path approach suffers from the exact same problems.

Instead, a capacity-constrained minimum s-t cut is likely a better fit for the two-dimensional case because dropping the capacity constraint gives a polynomially solvable minimum s-t cut problem.

The minimum s-t cut problem: Given a graph $G = (V, E)$, a source $s \in V$, a sink $t \in V$ and a weight function $w_1 : E \to \mathbb{R}_{\geq 0}$, the minimum s-t cut problem seeks for a partition of the nodes V into two sets U and $U^C = V \setminus U$ such that $s \in U$ and $t \in U^C$ and $\sum_{e \in \partial U} w_1(e)$ is minimized where $\partial U = \{\{v_i, v_j\} \in E \mid v_i \in U,\ v_j \in U^C\}$.

The capacity-constrained minimum s-t cut problem: Given a graph $G = (V, E)$, a source $s \in V$, a sink $t \in V$, two weight functions $w_1 : E \to \mathbb{R}_{\geq 0}$ and

$w_2 : E \to \mathbb{R}_{\geq 0}$ as well as a bound $B > 0$, the capacity-constrained minimum s-t cut problem seeks for a partition of the nodes V into two sets U and $U^C = V \setminus U$ such that $s \in U$ and $t \in U^C$ and $\sum_{e \in \partial U} w_1(e)$ is minimized and $\sum_{e \in \partial U} w_2(e) \leq B$ holds.

One can reformulate the problem (TR-IP) as a capacity-constrained minimum s-t cut problem on a graph $G = (V, E)$, which searches for a minimum s-t cut C with regards to a weight function $w_1 : E \to \mathbb{R}_{\geq 0}$ and adheres to a capacity constraint $w_2(C) \leq \Delta$ with a capacity consumption function $w_2 : E \to \mathbb{R}_{\geq 0}$. The graph construction is derived from [99] which tackles a similar problem without a capacity constraint which in our case is modelled by w_2. The idea of using a minimum cut approach for energy minimization is common in image segmentation, see e.g. [18]. The construction of the minimum s-t cut problem after dropping the capacity constraint mirrors the one in [99]. The minimum s-t cut problem is well-studied and can for example be solved as a max-flow problem with the Ford–Fulkerson algorithm, see [63] pp. 178-182. This also shows that the inner problem of the Lagrangian relaxation problem (LR-Δ) is polynomially solvable for a fixed Lagrange multiplier via this approach, see Section 4.2 for the problem (LR-Δ).

We first describe the graph construction and then show that a solution of the capacity-constrained minimum s-t cut problem is equivalent to a solution of (TR-IP).

The graph contains $N \times M \times (|\Xi| - 1) + 2$ nodes which include a source s and a sink t. Excluding the source and the sink, all nodes are of the form $v = (i, j, \ell) \in [N] \times [M] \times [|\Xi| - 1]$. To access the entries of a node $v = (i, j, \ell) \in V \setminus \{s, t\}$, we define the notation

$$row(v) = i, \;\; col(v) = j, \;\; \text{and} \;\; lvl(v) = \ell.$$

For s we set $lvl(s) = 0$ and for t we set $lvl(t) = |\Xi| = m$. Note that we do not assign a *row* and *col* value for the source and the sink. In the graph we consider two different types of edges.

Horizontal Edges in $G = (V, E)$: Two nodes $v_i, v_j \in V \setminus \{s, t\}$ are connected by a horizontal edge if

$$lvl(v_i) = lvl(v_j)$$

and either

$$row(v_i) = row(v_j) \text{ and } |col(v_i) - col(v_j)| = 1$$

or

$$|row(v_i) - row(v_j)| = 1 \text{ and } col(v_i) = col(v_j)$$

hold. Each edge has an edge weight $w_1(v_i, v_j) = \alpha |\xi_{lvl(v_i)+1} - \xi_{lvl(v_i)}|$. The capacity consumption of the edge is given by $w_2(v_i, v_j) = 0$. The horizontal edges model the total variation term in our objective.

Vertical Edges in $G = (V, E)$: Two nodes $v_i, v_j \in V \setminus \{s, t\}$ are connected by a vertical edge if

$$row(v_i) = row(v_j)$$

and

$$col(v_i) = col(v_j)$$

and

$$|lvl(v_i) - lvl(v_j)| = 1$$

hold. The weight is given by

$$w_1(v_i, v_j) = c_{row(v_i),col(v_i)}(\xi_{\max\{lvl(v_i),lvl(v_j)\}} - u_{row(v_i),col(v_i)}) + C$$

where

$$C \coloneqq (\xi_m - \xi_1)(\max_{i\in[N],j\in[M]} c_{i,j} + 5\alpha) \tag{5.3}$$

is a constant offset to ensure that all vertical edge weights are nonnegative and a minimum s-t cut contains exactly NM vertical edges which we prove in Lemma 5.5. The capacity consumption is given by

$$w_2(v_i, v_j) = |\xi_{\max\{lvl(v_i),lvl(v_j)\}} - u_{row(v_i),col(v_i)}|.$$

Furthermore, a vertical edge from s to $v_i \in V \setminus \{s, t\}$ exists with weight

$$w_1(s, v_i) = c_{row(v_i),col(v_i)}(\xi_1 - u_{row(v_i),col(v_i)}) + C$$

and capacity consumption

$$w_2(s, v_i) = |\xi_1 - u_{row(v_i),col(v_i)}|$$

if $lvl(v_i) = 1$. Finally, a vertical edge from $v_j \in V \setminus \{s, t\}$ to t exists with weight

$$w_1(v_j, t) = c_{row(v_j),col(v_j)}(\xi_m - u_{row(v_j),col(v_j)}) + C$$

and capacity consumption

$$w_2(v_j, t) = |\xi_m - u_{row(v_j),col(v_j)}|$$

if $lvl(v_j) = m - 1$. The vertical edges describe the choice of Ξ for the corresponding entry in $d + u$. An example of the graph construction is illustrated in Figure 5.3.

We now show that there is an equivalence between a capacity-constrained minimum s-t cut on G and a solution of (TR-IP). For the proof we first need to show the following lemma.

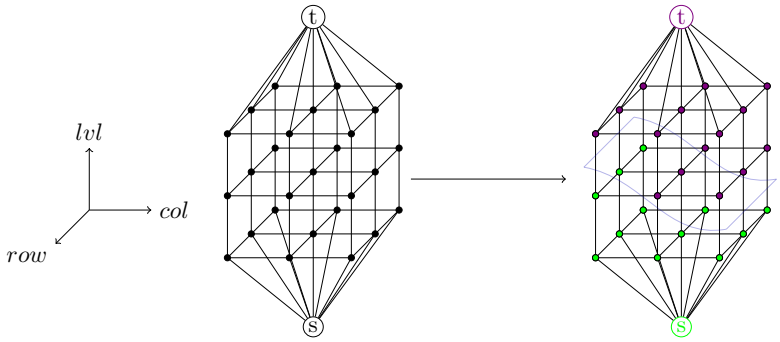

Figure 5.3: Graph $G = (V, E)$ with $N = M = 3$ and $|\Xi| = 4$. A possible cut is illustrated in blue.

Lemma 5.5. *Every s-t cut contains at least NM* ***vertical*** *edges. Let S be a capacity-constrained minimum s-t cut on G. Then S contains exactly NM* ***vertical*** *edges. For two* ***vertical*** *edges $e_1 = \{v_i, v_j\}$ and $e_2 = \{v_k, v_l\}$ in S it holds that either $row(v_i) \neq row(v_k)$ or $col(v_i) \neq col(v_k)$.*

Proof. The graph contains NM disjoint paths from s to t, which only contain vertical edges. Thus, any s-t cut has to contain at least NM vertical edges. Assume that an s-t cut S contains more than NM vertical edges. Then there must exist at least one row-column-combination for which at least two vertical edges are cut. We now add all horizontal edges containing a node with this row-column-combination to S (if they are not already in the cut) and remove all but the cheapest vertical edge of said row-column-combination. Obviously, this cut still separates s and t. There are at most $4(m-1)$ horizontal edges added to the cut, inducing an additional total cost of at most $4\alpha|\xi_m - \xi_1|$. Thus, we have decreased the cost of S. Furthermore, the capacity constraint holds as long as it did for our original S, as the horizontal edges have a capacity consumption of 0 while the vertical edges have a non-negative capacity consumption. It follows that our original S was not a capacity-constrained minimum s-t cut on G. □

Lemma 5.6. *Let S be a capacity-constrained minimum s-t cut on G. Let $e_1 = \{(\bar{i}, \bar{j}, \bar{\ell}), (\bar{i}, \bar{j}, \bar{\ell}+1)\}$ and $e_2 = \{(\tilde{i}, \tilde{j}, \tilde{\ell}), (\tilde{i}, \tilde{j}, \tilde{\ell}+1)\}$ be two* ***vertical*** *cut edges with $\bar{i} = \tilde{i}$, $|\bar{j} - \tilde{j}| = 1$ and $\bar{\ell} \leq \tilde{\ell}$. Then S contains exactly $|\bar{\ell} - \tilde{\ell}|$* ***horizontal*** *edges between nodes with the row-column-combination $(\bar{i}, \bar{j})$ and nodes with the row-column-combination $(\tilde{i}, \tilde{j})$ with the level entries in $(\bar{\ell}, \tilde{\ell}] \cap \mathbb{N}$.*

Proof. Let S be a capacity-constrained minimum s-t cut on G. We show that S contains at least all the horizontal edges between nodes with the row-column-combination $(\bar{i}, \bar{j})$ and nodes with the row-column-combination $(\tilde{i}, \tilde{j})$ with level entries in $(\bar{\ell}, \tilde{\ell}] \cap \mathbb{N}$ but does not need to contain more to be an s-t cut. Because the horizontal edges have a positive weight, this already means that no more horizontal edges can be in a minimum s-t cut.

If S does not contain all of the aforementioned horizontal edges, then we can find an edge $\bar{e} = \{(\bar{i}, \bar{j}, \ell^*), (\tilde{i}, \tilde{j}, \ell^*)\}$ with $\ell^* \in (\bar{\ell}, \tilde{\ell}] \cap \mathbb{N}$, which is not in S. We can construct a path $\bar{p}$ which contains the vertical edges from s to $(\tilde{i}, \tilde{j}, \ell^*)$, the horizontal edge $\bar{e}$ and the vertical edges from $(\bar{i}, \bar{j}, \ell^*)$ to t. This s-t path does not contain any edge in S due to Lemma 5.5, but this is a contradiction to the statement that S is a minimum s-t cut. Thus, all the aforementioned horizontal edges have to be contained in S to ensure that no path from s to t remains uncut. The situation is visualized in Figure 5.4 (a) and (b).

We now show that no further horizontal edges are needed for S to be an s-t cut. Let $\bar{S}$ be a subset of S which only contains the aforementioned vertical and horizontal edges. If $\bar{S}$ is not an s-t cut without additional horizontal edges, then we can find a path $p = \{s, v_1, \ldots, v_o, t\}$ from s to t with a minimal number of elements where no element is also in $\bar{S}$ (potential cut edges). The path p needs to contain at least one horizontal edge because for all direct s-t paths a vertical edge is included in $\bar{S}$. Without loss of generality let $e = \{(i, j, \ell), (i, j+1, \ell)\}$ be an horizontal edge in p. Then for both row-column-combinations the vertical edges in $\bar{S}$ are of the form $\bar{e}_1 = \{(i, j, \ell_1), (i, j, \ell_1 + 1)\}$ and $\bar{e}_2 = \{(i, j+1, \ell_2), (i, j+1, \ell_2 + 1)\}$ with $\ell_1, \ell_2 < \ell$ or $\ell_1, \ell_2 \geq \ell$ (otherwise the horizontal edge would be in $\bar{S}$). In the first case we could reduce the number of elements in p by directly following the vertical edges from (i, j, ℓ) or $(i, j+1, \ell)$ to t which is a contradiction to our assumption of minimality. The same holds for the second case just with the role of t replaced by s. Thus, p does not contain any horizontal edges. In turn, p does not connect s and t which means that $\bar{S}$ is already an s-t cut. Because all edges have a positive weight, the optimality of the cut implies that no further horizontal edges can be included in a mimimum s-t cut. So S is identical to $\bar{S}$. □

Remark 5.7. Obviously the previous lemma also holds for the case that $|\bar{i} - \tilde{i}| = 1$ and $\bar{j} = \tilde{j}$.

Theorem 5.8. *Let S be a capacity-constrained minimum s-t cut on G. Then we can construct an optimal point for* (IP) *from the cut. The objective values differ by exactly NM times the offset constant C. Let $(d, \delta, \beta, \gamma)$ be optimal for* (IP)*, then we can derive a capacity-constrained minimum s-t cut on G with the same objective value offset by NM times the constant C from this solution.*

Proof. We first show that we can construct feasible points for (IP) from capacity-constrained minimum s-t cuts and capacity-constrained s-t cuts from optimal points where the respective objective values only differ by a constant.

Let S be a capacity-constrained minimum s-t cut on G. For each of the NM row-column-combinations exactly one vertical edge is cut, see Lemma 5.5. We now construct a solution $(d, \delta, \beta, \gamma)$ for (IP) from the cut. For each row-column-combination we can find a cut vertical edge $e = \{v_k, v_l\}$ which corresponds to choosing $d_{row(v_k),col(v_k)}$ such that $d_{row(v_k),col(v_k)} + u_{row(v_k),col(v_k)} = \xi_{\max\{lvl(v_k),lvl(v_l)\}}$. We

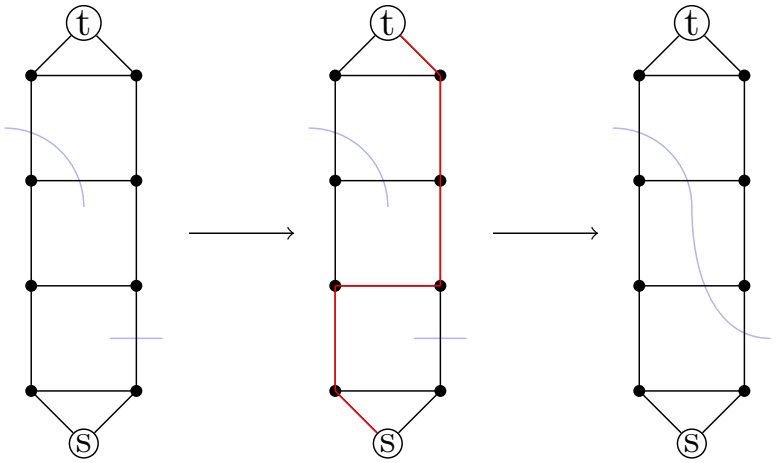

Figure 5.4: Visualisation of Lemma 5.6: If the cut does not contain all horizontal edges between the vertical cut edges then we can find a path connecting s and t which is colored in red in the figure.

set $\delta_{i,j} = |d_{i,j}|$ for all $i \in [N], j \in [M]$. These values coincide with the capacity costs of the cut edges by construction of the graph. For the terms modeling the total variation we set $\beta_{i,j} = |u_{i+1,j} + d_{i+1,j} - u_{i,j} - d_{i,j}|$ for all $i \in [N-1], j \in [M]$ and $\gamma_{i,j} = |u_{i,j+1} + d_{i,j+1} - u_{i,j} - d_{i,j}|$. It follows from the previous lemma, that the values are exactly the sum of the weights of the cut horizontal edges. It holds that $u+d$ only takes values in $\Xi^{N\times M}$ because each horizontal layer in the graph represents a value in Ξ. Because S is a capacity-constrained minimum s-t cut, the capacity constraint on the graph is fulfilled and $\sum_{i,j} \delta_{i,j} \leq \Delta$ holds. Thus, $(d, \delta, \beta, \gamma)$ is feasible for (IP). The objective value of S is given by

$$\sum_{e=\{v_k,v_l\}\in S_v} (c_{row(v_k),col(v_k)}(\xi_{\max\{lvl(v_k),lvl(v_l)\}} - u_{row(v_k),col(v_k)}) + C) + \sum_{e\in S_h} w_1(e)$$

where S_v is the set of vertical edges and S_h is the set of horizontal edges in S. From the construction of d it follows that

$$\sum_{e=\{v_k,v_l\}\in S_v} c_{row(v_k),col(v_k)}(\xi_{\max\{lvl(v_k),lvl(v_l)\}} - x_{row(v_k),col(v_k)} = \sum_{i,j} c_{i,j} d_{i,j}$$

holds. As stated the sum of the weights of the horizontal cut edges is identical to the sum of the $\beta_{i,j}$ and $\gamma_{i,j}$ so

$$\sum_{e\in S_h} w_1(e) = \alpha \left(\sum_{j=1}^{M} \sum_{i=1}^{N-1} \beta_{i,j} + \sum_{i=1}^{N} \sum_{j=1}^{M-1} \gamma_{i,j} \right).$$

The objective value of $(d, \delta, \beta, \gamma)$ for (IP) differs from the objective value of S for the capacity-constrained minimum s-t cut problem by exactly $|S_v|C = NMC$ due to Lemma 5.5.

We now show that we can construct a capacity-constrained s-t cut $\bar{S}$ on G from an optimal point $(d^*, \delta^*, \beta^*, \gamma^*)$ of (IP). For $i \in [N]$ and $j \in [M]$ we define ℓ such that $d^*_{i,j} + u_{i,j} = \xi_\ell$ holds. We add the edge between the nodes $(i, j, \ell - 1)$ and (i, j, ℓ) to $\bar{S}$ if $1 < \ell < m$. If $\ell = 1$, we add the edge between s and $(i, j, 1)$, and if $\ell = m$, we add the edge between $(i, j, m-1)$ and t to $\bar{S}$ instead. The capacity consumption of the vertical edges are exactly $\delta^*_{i,j}$, which ensures that the capacity constraint on the graph follows because the capacity constraint is fulfilled for (IP) for the given δ^*.

For $i \in [N-1]$ and $j \in [M]$ we set ℓ_1 such that $d^*_{i,j} + u_{i,j} = \xi_{\ell_1}$ and ℓ_2 such that $d^*_{i+1,j} + u_{i+1,j} = \xi_{\ell_2}$. We add all the horizontal edges e to $\bar{S}$ with $e = \{(i, j, k), (i + 1, j, k)\}$ where $k \in \{\min\{\ell_1, \ell_2\}, \dots, \max\{\ell_1, \ell_2\} - 1\}$. These edges correspond to the $\beta^*_{i,j}$.

For $i \in [N]$ and $j \in [M-1]$ we set ℓ_1 such that $d^*_{i,j} + u_{i,j} = \xi_{\ell_1}$ and ℓ_2 such that $d^*_{i,j+1} + u_{i,j+1} = \xi_{\ell_2}$. We add all the horizontal edges e to $\bar{S}$ with $e = \{(i, j, k), (i, j + 1, k)\}$ with $k \in \{\min\{\ell_1, \ell_2\}, \dots, \max\{\ell_1, \ell_2\} - 1\}$. These edges correspond to the $\gamma^*_{i,j}$.

The objective value of $\bar{S}$ is given by

$$\begin{aligned}
&\sum_{e=\{v_k, v_l\} \in \bar{S}_v} (c_{row(v_k), col(v_k)}(\xi_{\max\{lvl(v_k), lvl(v_l)\}} - x_{row(v_k), col(v_k)}) + C) + \sum_{e \in \bar{S}_h} w_1(e) \\
&= \sum_{i,j} c_{i,j} d_{i,j} + |\bar{S}_v| C + \alpha \left(\sum_{j=1}^{M} \sum_{i=1}^{N-1} \beta_{i,j} + \sum_{i=1}^{N} \sum_{j=1}^{M-1} \gamma_{i,j} \right)
\end{aligned}$$

where $\bar{S}_v$ is the set of vertical edges and $\bar{S}_h$ is the set of horizontal edges in $\bar{S}$.

We now show optimality. Let S^* be an optimal capacity-constrained minimum s-t cut. Then we can construct a feasible point for (IP). The objective value is exactly NMC below the objective value of S^*. Assume that the constructed point is not optimal. Then we can find an optimal point with an objective value strictly smaller. We can now construct a capacity-constrained s-t cut from this optimal point with an objective value exactly NMC larger. This means that the constructed cut has a lower objective value than S^* which is a contradiction to the optimality of S^*. This shows the optimality of the feasible points constructed from minimum s-t cuts. The same argumentation can be used to show that the capacity-constrained s-t cuts derived from optimal point of (IP) are minimal. □

Remark 5.9. This equivalence to solutions of (IP) also allows us to construct a solution to (TR-IP).

Remark 5.10. By choosing $w(e) = w_1(e) + \mu w_2(e)$ as the weight function and dropping the capacity function for the constructed graph we can solve the inner problem of the Lagrangian problem (LR-Δ) with a fixed multiplier $\mu \geq 0$ as a minimum s-t cut problem.

Capacity-constrained global minimum cuts can be calculated efficiently as bicriteria minimum cuts as detailed in Theorem 2.4 of [3]. For our problem formulation we need a formulation with s and t such that this approach cannot be used in general. For very special cases like $u \equiv 0$ and $c \geq 0$ the problem (TR-IP) (for which 0 is the smallest value in Ξ) can be solved as a capacity-constrained global minimum cut problem but it is straightforward to see that an optimal solution is always given by $d \equiv 0$. The bicriteria s-t minimum cut problem, however, is NP-hard in general as shown in Theorem 6 of [81] but the proof of NP-hardness does not extend to grid graphs.

5.3 The knapsack problem

The knapsack problem is a well-known combinatorial problem. The problem is proven to be NP-hard but admits a pseudo-polynomial algorithm, see [63].

The knapsack problem: Given a finite set of items I, a value function $v : I \to \mathbb{R}_{\geq 0}$, a weight function $w : I \to \mathbb{R}_{\geq 0}$ and a budget $B > 0$, the knapsack problem seeks for a subset U of I such that the sum $\sum_{i \in U} v(i)$ is maximized and the budget constraint $\sum_{i \in U} w(i) \leq B$ is fulfilled.

In Chapter 6 we will use a reduction from the knapsack problem to prove NP-hardness for the one-dimensional problem (1D-w). The integer linear programming formulation of the knapsack problem is given by

$$\text{(KS)} \qquad \begin{aligned} \max_x \sum_{i \in I} v(i) x_i \\ \text{s.t.} \sum_{i \in I} w(i) x_i \leq B \\ x_i \in \{0, 1\} \text{ for } i \in I \end{aligned}$$

where $v(i), w(i)$ are the given parameters for all $i \in I$.

The linear programming relaxation can be solved with a greedy algorithm, see [32]. The items are ordered based on the ratio between the value and the weights. The problem is solved by setting all variables to 1 which correspond to items with the best ratio such that the sum of the weights of the items is at most B but the weight of the next best item with regard to this ratio exceeds the remaining budget. The

variable corresponding to this final item is set as the ratio of the remaining budget and the weight of the item. Thus, the solution to the linear programming relaxation contains only integer values except for one entry. We will see in Chapter 7 that a similar property holds for (IP) and the corresponding linear programming relaxation (LP).

A well known class of cutting planes for the knapsack problem are the cover inequalities, see e.g. [101]. Let $U \subset I$ be a subset such that $\sum_{i \in U} w(i) > B$ and $\sum_{i \in U \setminus \{j\}} w(i) < B$ for every $j \in U$ hold, then

$$\sum_{i \in U} x_i \leq |U| - 1$$

is a valid inequality for the knapsack polyhedron. This concept can be used directly for the capacity constraint of (1D-w) as the left hand coefficients are positive integer values. Applying the concept to the optimization variable d for (TR-IP) does not improve the formulation as all coefficients in the capacity constraint have absolute value of 1 and thus any cover inequality is already fulfilled if the capacity constraint holds. For a non-uniform discretization and thus weights different from 1 this concept could prove useful but is beyond the scope of this dissertation.

An extension of the knapsack problem is the two-dimensional knapsack problem in which the items are rectangular shaped and have to be fit into a rectangular domain/knapsack, see [26]. This problem is closer to the two-dimensional (TR-IP) with a binary control set and previous control $u \equiv 0$ but would need to allow exponentially many items with shapes induced by subgraphs of the grid, possibly with holes, and would also need additional constraints to ensure that two chosen items do not overlap.

5.4 The minimum bisection problem

The minimum bisection problem will be useful in different ways in the later chapters. In Chapter 6 we will conjecture the NP-hardness of (TR-IP) based on a reduction from the *minimum bisection problem* on grid graphs with an arbitrary number of holes. In the outlook we will discuss how the insights may become useful for future research.

The minimum bisection problem (MBP): Given a graph $G = (V, E)$, (MBP) seeks a partition into two sets $S, V \setminus S$ such that $|S|, |V \setminus S| \leq \lceil |V|/2 \rceil$ which minimizes the cardinality of the set of cut edges $C \coloneqq \{\{v_i, v_j\} \in E \mid v_i \in S,\ v_j \in V \setminus S\}$. $|C|$ is called the bisection width.

In [80] and [41] the minimum bisection problem on solid grid graphs is discussed and polynomial algorithms are presented. In the earlier paper [80] an $\mathcal{O}(|V|^5)$ algorithm is given and in [41] the run time is improved to $\mathcal{O}(|V|^4)$. In this section we want to derive an $\mathcal{O}(|V|^2)$ algorithm to determine the so-called minimum cut ratio on solid grid graphs. To this end we briefly restate the results from [80] and [41] regarding the shapes of optimal cuts in a solid grid graph. In Lemma 1 of [80] it is shown that the optimal bisection of a solid grid graph is a set of disjoint simple cuts. Simple cuts are defined as cuts which correspond to cycles in the dual graph that contain the infinite face. Thus, a simple cut corresponds to a path in the dual graph if the infinite face is removed, see [80]. Furthermore, it is shown in Lemma 4 in [80] that each of the simple cuts has to be optimal itself. For further details regarding the definition of optimality of simple cuts and proofs for the statements we refer the reader to [80] and note that from our understanding the formula $B(G') - m - n - 3 + R(C) - L(C)$ given in the proof of Lemma 4 has to be corrected to $B(G') - m - n + 3 + R(C) - L(C)$ for the correctness of the proof.

The authors in [41] concern themselves with the problem of finding a minimum partition of a solid grid graph where one of the sets has size k based on the results from [80]. This is called a k cut.

Definition 5.11. The set of cut edges of a simple cut is called a **segment**.

The authors in [41] argue based on the results from [80] that there exists a k cut only consisting of simple cuts with certain corresponding segments which they refer to as either *straight lines, corners, stairs, clamps or squares*. They, however, do not go into further detail to prove this result but only give a proof sketch and argue that this directly follows from Lemma 3 and 4 from [80]. We will first present the definitions and segment types given in [41] restricted to simple cuts and give an extensive proof that we can find a k cut such that the corresponding segments are either a straight line, a corner, a stair, a clamp or a square. We use this insight to obtain an $\mathcal{O}(|V|^2)$ algorithm to obtain the so-called minimum cut ratio. We use the ideas from [41] but changed the following definitions such that we only work on the edges of the original graph.

Definition 5.12. Let $G = (V, E)$ be a solid grid graph, D the dual graph and C a simple cut with a path p in D and a segment s in E where the cut edges are given in the order in which they are cut by C.

A **bend** consists of one horizontal edge $e_1 \in s$ and one vertical edge $e_2 \in s$ which share a common incident node $v \in V$. We say that the bend is oriented to the top-left if e_1 is to the left of v and e_2 is to the top of v. The orientations bottom-left, top-right and bottom-right are defined in the same manner. A bend pointing towards the top-left points in a opposite direction of a bend pointing towards the bottom-right. A bend pointing towards the top-right points in a opposite direction of a bend pointing

towards the bottom-left. If two bends do not point in opposite directions, they share a direction.

A **break** is an edge contained in two bends which point in opposite directions and share an edge.

Let q be a sub-path of p. Let b denote the set of cut edges corresponding to the path q in the order in which they are cut. If b only contains vertical or horizontal edges and the first edge is also the first edge or is part of a bend in s and the last edge is also the last edge or is part of a bend in s, we call b a **bar**. If the first or last edge in b is the first or last edge in s, we say that the bar ends in the infinite face f_∞. If the first or last edge in b is part of a bend in s, we say that it ends in the bend. Two bends are **consecutive** if there exists a bar in s ending in both.

A subset b of s corresponding to q is called a **broken bar** if it consists of a break and two bars which each end in a different bend of the break.

The above definitions define the framework to characterise the different types of segments and the segments are visualized in Figure 5.5.

Definition 5.13. A segment s is called

- a straight segment if it contains no bend.
- a corner segment if it contains exactly one bend.
- a stair segment if any consecutive bends of s point in opposing directions. If s contains no breaks it has exactly two bends. Otherwise it has a broken bar b such that $s \setminus b$ contains at most two bars each ending in f_∞.
- a clamp segment if there are two bends f_1 and f_2 of s pointing in a common direction and s can be partitioned into a (broken) bar b ending in f_1 and f_2 and two bars ending in f_∞ and f_1 or f_2 respectively.
- a square segment if there are three bends f_1, f_2, f_3 of s such that s can be partitioned into a (broken) bar b ending in f_1 and f_2, a bar b' ending in f_2 and f_3 and two bars ending in f_∞ and f_1 or f_3 respectively. The pairs f_1, f_2 and f_2, f_3 each point in a common direction. In case that b is a broken bar. f_1 and the consecutive bend in the break point in a common direction. If b is a bar, then its length is equal to the length of b' or smaller by 1. If b is a broken bar, then its length is equal to the length of b' or larger by 1.

If a stair, clamp or square contains a broken bar, we say that it has a break.

In the following we will turn to the dual graph which we denote by D for a grid graph G, whereas D^- is the graph we obtain after removing the node representing the infinite face from D. The graph D^- can embedded in an infinite grid which we refer to as the infinite dual grid to distinguish it from the infinite grid in which

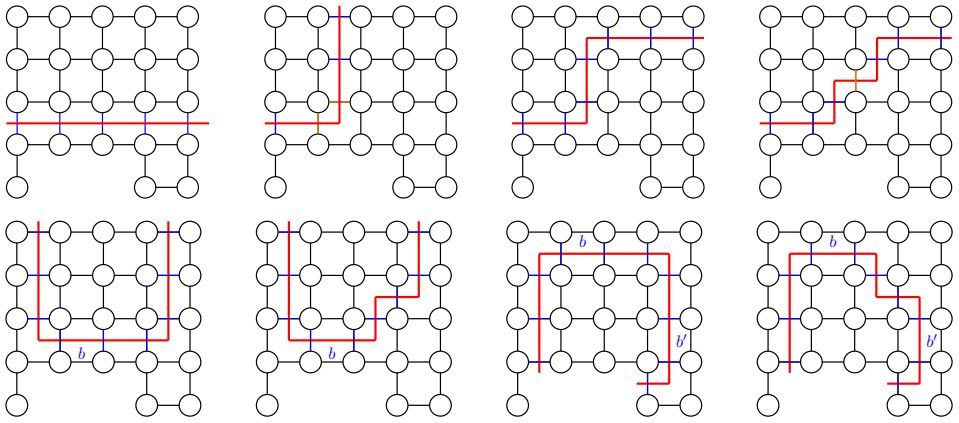

Figure 5.5: Visualized are the optimal segments from Definition 5.13. The red lines denote the paths in the dual graph, while the segments are the blue (and orange) edges cut by the paths. In the second figure we have marked a bend pointing to the bottom-right in orange while we have marked a break (inside a broken bar) in orange in the fourth figure. We note that these are not the only bends and breaks. Shown are the following segments.
Top row: straight, corner, stair w/o break, stair w/ break.
Bottom row: clamp w/o break, clamp w/ break, square w/o break, square w/ break.

the graph G lies. Let B denote the set of nodes in the infinite dual grid which are adjacent to a node in D^- but not in D^- itself. We obtain the graph D^+ by adding the nodes in B and the edges of the dual grid connecting nodes in D^- to nodes in B to the graph D^-.

The original idea from [80] to derive optimal structures was to inspect the optimal path in D^+ between two nodes in B such that the number of nodes on one side is of a given size. In the proof of the following theorem we will use this argumentation.

Before we turn to the theorem, we need an additional insight. Let C be a simple cut in a solid grid graph G with a corresponding segment s which partitions the nodes of the graph into two sets S, S^C. Let β be a bend in the segment with the edges e_1, e_2 pointing outwards of the set S. These two edges have a common incident node v, which has two more incident edges e_3, e_4 in the infinite grid. We can replace the edges e_1, e_2 by the edges e_3, e_4 in the segment to obtain a new valid cut and bend. The new cut partitions the graph into two sets $\tilde{S}, \tilde{S}^C$ such that $|S| = |\tilde{S}| + 1$ if $v \in S$ or $|S| = |\tilde{S}| - 1$ if $v \notin S$ (in either case the sets S and $\tilde{S}$ are identical except for the node v). There are two possible cases. In the first case the edges e_3, e_4 were both part of the graph G. Then the number of cut edges remains the same. In the latter case at least one edge is not part of G and the number of cut edges is reduced accordingly. We call this procedure *inner fold* if $v \in S$ and *outer fold* if $v \notin S$. The folds are visualized in Figure 5.6.

We can also change the position of the break in a segment by applying outer folds to the broken bar b and inner folds to a bar b' until the broken bar is transformed into a bar, see Figure 5.7. The bar b' may or may not be a broken bar afterwards.

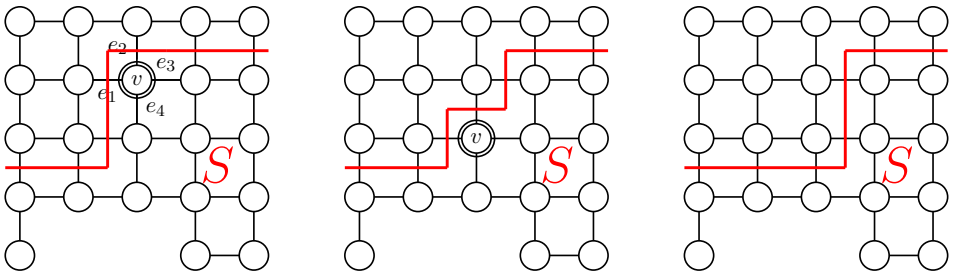

Figure 5.6: Visualized from left to right is the application of two inner folds successively. Since an outer fold is the inverse application, two outer folds can be see from right to left.

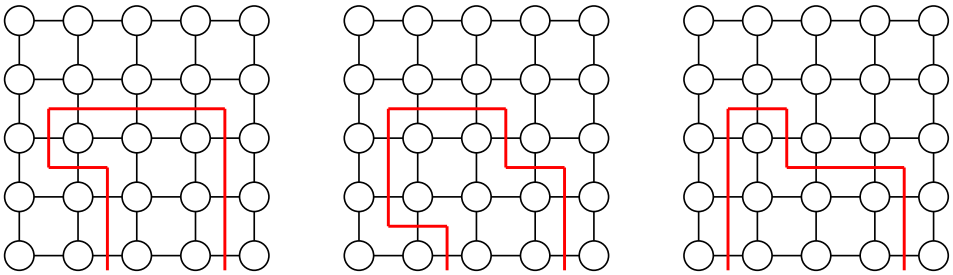

Figure 5.7: The break is moved from one (broken) bar to another bar. The former broken bar is now a bar and the former bar is now a broken bar.

Theorem 5.14. *For every solid grid graph and any* $0 < k \leq \frac{|V|}{2}$ *there exists an optimal* k *cut which consists of only disjoint simple cuts and each corresponding segment is of a type described in Definition 5.13.*

Proof. Let C be an arbitrary optimal k cut. We already know that C consists of finitely many disjoint simple cuts, see Lemma 1 in [80]. Let the simple cuts be denoted by $C_1, \ldots, C_\ell$. We apply all but one of the simple cuts to the original grid graph to obtain a new grid graph, meaning we consider the cut we obtain from the combination of all cuts $C_1, \ldots, C_\ell$ except for the simple cut C_i. We choose the graph for which the remaining simple cut C_i has a cycle in the dual graph as each of the simple cuts creates a new additional solid grid graph by cutting off a part of the original grid graph. The remaining simple cut C_i thus is a k_i cut in the new grid graph G. We now show that the segment of this simple cut is of a type described in Definition 5.13 or we can find an equally good k_i cut with a corresponding segment of a type described in Definition 5.13. Since we have choosen an arbitrary C_i, this suffices to prove the result.

The cut C_i has a corresponding cycle in the dual graph D of G which contains the infinite face, see [80]. This corresponds to a path P_i in the graph D^+ starting in one node in B and ending in a different node in B. Let these two nodes in B be denoted by $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$ where x_j, y_j for $j \in \{1, 2\}$ are the coordinates in the infinite dual grid. We assume without loss of generality that $x_1 \leq x_2$ and $y_1 \leq y_2$. (Otherwise we can obtain this by rotating or mirroring the original graph.) Since the remaining path lies completely in D^-, we know that there are at least $(x_2 - x_1) + (y_2 - y_1)$ cut edges for C_i.

We can construct two shortest paths between the nodes in the infinite dual grid. The first path P_a starts in (x_1, y_1), goes vertically to (x_1, y_2), and then horizontally to (x_2, y_2). The other path P_b starts in (x_1, y_1), goes horizontally to (x_2, y_1), and then vertically to (x_2, y_2). Let the cuts be denoted by C_a, C_b. C_a and C_b split the set of nodes into two sets respectively which are denoted by S_{P_j} and $S^C_{P_j}$ for $j \in \{a, b\}$. Without loss of generality we assume that $S_{P_a} \subset S_{P_b}$, otherwise we switch the roles. We note that these cuts could cross one of the other cuts in C but the following construction ensures that the final cut does not cross other cuts of C and is a simple cut so we can ignore this case. We now differentiate between three cases.

Case 1: Let $|S_{P_a}| \leq k_i \leq |S_{P_b}|$ hold. Because C_i is a simple cut, the amount of cut edges has to be at least $(x_2 - x_1) + (y_2 - y_1)$. In the case that at least one of the two inequalities holds with equality, we have already found a cut with a corresponding straight segment (if $x_1 = x_2$ or $y_1 = y_2$) or corner segment which has at most $(x_2 - x_1) + (y_2 - y_1)$ cut edges with either C_a or C_b, where it is ensured that the corresponding path lies completely in the dual graph D due to the optimality of C_i (otherwise, we would have found a k_i cut with a smaller number of cut edges by tracing the boundary for the parts of the path outside of D^-).

In the other case, we can find a path between $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$ such that the set of nodes is partitioned into two sets with one of the sets having size k_i by starting with the path P_b and using inner folds along the y dimension, see Figure 5.8. We start with set S_{P_b} and each inner fold either does not change the set S_{P_b} if the folded node is not in G, or decreases the number of nodes by 1 if the node is in G. Since we obtain the path P_a after finitely many folds and $|S_{P_a}| < k_i$, we ensure that we obtain a k_i cut. The constructed path has at most $(x_2 - x_1) + (y_2 - y_1)$ cut edges. Due to the optimality of C_i, C_i and the newly constructed cut both have the same amount of cut edges given by $(x_2 - x_1) + (y_2 - y_1)$, which ensures that the path corresponding to the newly constructed cut does not contain the infinite face (otherwise, the cut would have less cut edges than C_i). Thus, from this construction we obtain a simple cut which is a k_i cut and has the same amount of cut edges as C_i. The segment of the constructed cut is a stair segment, possibly with a break. Case 1 is visualized in Figure 5.8.

Case 2: This case is visualized in Figure 5.9. Let $|S_{P_b}| < k_i$ hold. We determine the lowest and highest x, y values for nodes $v \in D$ in the path corresponding to the cut C_i. Let

$$x_\ell \coloneqq \min\{x : (x, y) \in P_i\},\ x_r \coloneqq \max\{x : (x, y) \in P_i\}$$

and

$$y_b \coloneqq \min\{y : (x, y) \in P_i\},\ y_t \coloneqq \max\{y : (x, y) \in P_i\}.$$

We construct a new path P in the infinite dual grid. In Case 1 the paths P_a and P_b functioned as a bounding box. This new path P now fulfills this purpose.

We start in (x_1, y_1) then continue horizontally to (x_ℓ, y_1) and then to (x_ℓ, y_b) vertically. Afterwards, we continue horizontally to (x_r, y_b) and then vertically to (x_r, y_t). Finally, we continue horizontally to (x_2, y_t) and vertically to (x_2, y_2). This partitions the graph into two sets S_P and S_P^C such that $|S_P| \geq k_i$. We note that it is possible that $S_P^C = \emptyset$ if the whole path lies outside of D^- and only in the remaining infinite dual grid. The newly constructed path and the path P_i corresponding to C_i have the same length in the infinite dual grid.

We can now use inner folds along the shortest bar for which the corresponding subpath does not end in the infinite face (starting at the bend with a node with the highest x values for horizontal bars and at the bend with a node with the lowest y values for vertical bars). We continue this process along these resulting broken bars until the break is eliminated by the folds (or a k_i cut is obtained). By construction we do not fold a node $v \in S_{P_b}$. After finitely many folds we would obtain the path P_b. By the same argumentation as in the first case, we argue that we obtain a k_i cut as we start with the set S_P and each fold decreases the number of nodes by at most 1. If the resulting k_i cut is not a simple cut, this means that the number of cut edges is strictly smaller than the number of cut edges of the cut C_i, which is a contradiction to the optimality. Thus, the newly constructed cut is a simple cut which we refer to as C^*.

The corresponding segment to the constructed cut might not be of the type given in Definition 5.13 yet. Thus, we now perform operations to obtain a corresponding cut. The segment corresponding to the path P consists of up to 6 bars. By construction of C^* with path P^* the corresponding segment also consists of up to 6 bars, but one bar might now be a broken bar. This broken bar then corresponds to either the bar from (x_ℓ, y_b) to (x_r, y_b) or to the bar from (x_r, y_b) to (x_r, y_t) in P. Otherwise, depending on the position of the break we need to change the position of the broken bar as detailed in the introduction of the folds.

If (x_1, y_1) is not a leftmost node and (x_2, y_2) not a topmost node in the newly constructed path, we can shift the structure enclosed by the cut to the right and bottom until this is the case. Because C^* is a simple cut, for every node in S_{P^*} we can find a node in G to the right as otherwise the path P^* would contain the infinite

face at the corresponding spot. Thus, we can shift the structure one unit to the right and still obtain a k_i cut. The number of cut edges remains the same if the resulting cut is still a simple cut. Otherwise, it would decrease which would be a contradiction to the optimality. The same argumentation holds for the shift of one unit to the bottom. After this shifting operation, we now obtain a structure consisting of up to 4 bars where one bar might be a broken bar. The resulting segment is either a corner, stair, clamp or square, possibly with a break. To ensure that the lengths are correct, one can reduce the number of nodes along the side which is to short and instead envelope the corresponding number of nodes along the side which is too long while keeping the (broken) bar structures. (We always obtain a simple cut because otherwise we would obtain a cut with less cut edges which is a contradiction to the optimality of the original cut.) Again, case 2 is visualized in Figure 5.9.

Case 3: Let $|S_{P_a}| > k_i$ hold. This works the same way as case 2 with switched roles of the sets into which the graph is partitioned.

□

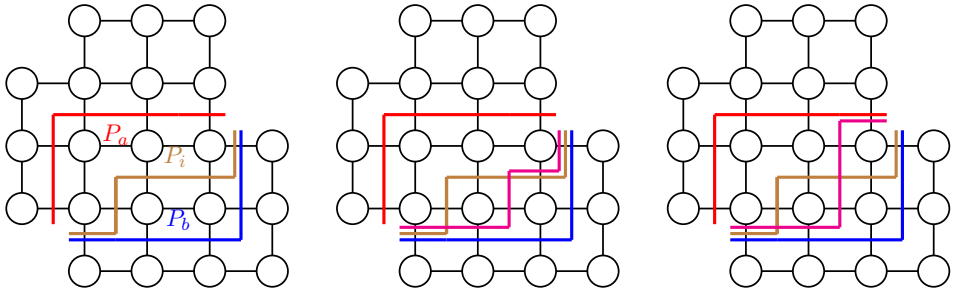

Figure 5.8: Visualized is the folding procedure in case 1. We start with the path P_b and fold along the y dimension until we obtain a cut (magenta) with the same number of nodes on each side as the cut C_i with path P_i.

We also see that for this k cut only containing simple cuts with segments given in Definition 5.13 no simple cut lies inside of the solid grid graph we have cut off with a different simple cut if inner folds can be applied to each. Otherwise, we could apply inner folds on both segments until one cut disappears or no inner fold can be applied, see Figure 5.10.

Definition 5.15. Let $G = (V, E)$ be a graph and C be a cut which partitions the nodes into U and U^C with the set of cut edges ∂U and $|U| \leq |U^C|$. The cut C is induced by the set U. The cut ratio of the cut C is given by $\frac{|\partial U|}{|U|}$. The minimum cut ratio of a graph $G = (V, E)$ is given by

$$\rho := \min_{U \subset V, 0 < |U| \leq \frac{|V|}{2}} \frac{|\partial U|}{|U|}.$$

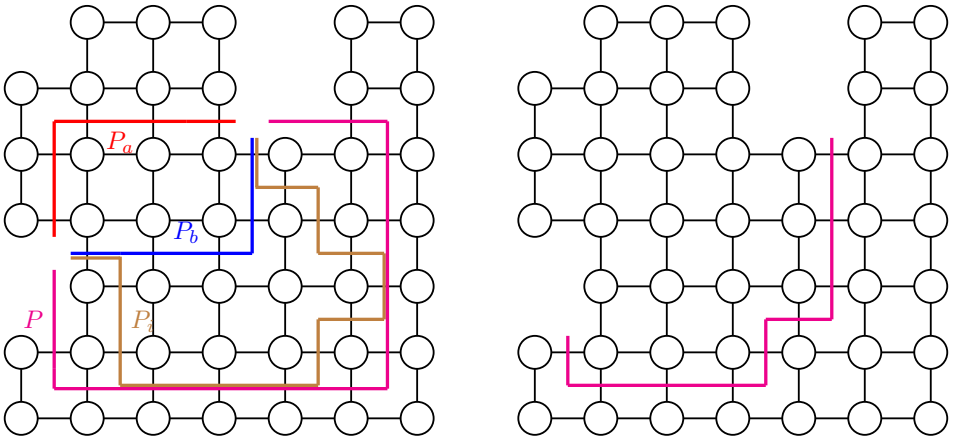

Figure 5.9: Visualization of case 2. On the left we can see P_a in red, P_b in blue, the path P_i corresponding to C_i in brown and the bounding path P in magenta which lies partially outside of the graph D^- (this shows that this path has less cut edges compared to the brown path). On the right we can see the path we obtain after applying the inner folds. The resulting segment is a clamp with a break. We note that the cut C_i with path P_i is not optimal here and the figure is only meant as a visualization of the procedure which does not require that the segment corresponding to C_i is of a type given in Definition 5.13.

Theorem 5.16. *Let $G = (V, E)$ be a solid grid graph. The minimum cut ratio*

$$\rho := \min_{U \subset V, 0 < |U| \leq \frac{|V|}{2}} \frac{|\partial U|}{|U|}$$

can be computed by just considering the cuts consisting of one segment of the types described in Definition 5.13.

Proof. Let C be an optimal k cut with $0 < k \leq \frac{|V|}{2}$ containing multiple simple cuts $C_1, \ldots, C_n$ (if C contains only one, then we are already done) which partitions the nodes into the sets U and U^C and realizes the minimum cut ratio. We know that we can assume that the cut consists only of segments described in Definition 5.13. Let the cut ratio $\frac{|\partial U|}{|U|}$ be denoted by ρ_C. We differentiate between two cases.

Case 1: The set of nodes cut off by the simple cuts has size k. Then every simple cut C_i in the cut C partitions the set of nodes into two sets U_i and U_i^C. It holds that $|U_i| \leq |U| = k$ and $\sum_i |\partial U_i| = |\partial U|$. Thus, there exists a simple cut C_i with a cut ratio at least as good as ρ_C.

Case 2: The set of nodes not cut off by simple cuts has size k. Without loss of generality let this set be denoted by U and the set of nodes cut off by U^C. We study the cut $\tilde{C}$ which we obtain from C by removing an arbitrary simple cut. The set of

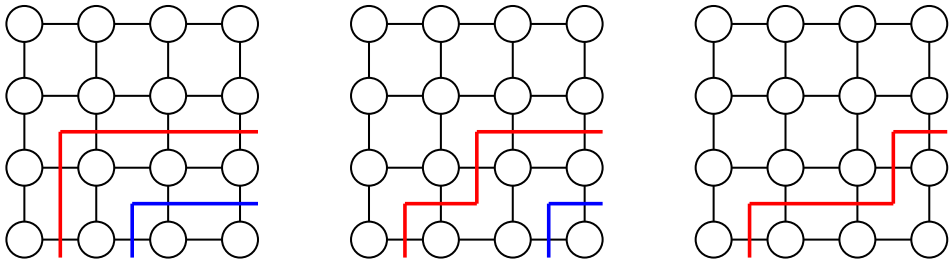

Figure 5.10: By applying inner folds to both simple cuts we can eliminate the inner cut.

nodes not cut off by this cut is a superset of the set of nodes not cut off by C with set size $k_2 > k$. If $\frac{|V|}{2} \geq k_2$ then we have obtained a k_2 cut with a strictly better cut ratio which is a contradiction to C realizing the minimum cut ratio. If $\frac{|V|}{2} < k_2$ then either the simple cut removed from C to obtain $\tilde{C}$ or $\tilde{C}$ itself has a ratio between cut edges and nodes cut off which is at least as good as the minimum cut ratio. This follow from $\frac{|\partial U^C|}{|U^C|} = \frac{|\partial U|}{|U^C|} \leq \frac{|\partial U|}{|U|}$ and the fact that either the simple cut or the cut $\tilde{C}$ must have a ratio between cut edges and nodes cut off which is at least as good as $\frac{|\partial U^C|}{|U^C|}$. In the former we have the case in the statement. In the later we obtain Case 1 with a $|V| - k_2$ cut.

□

Corollary 5.17. *The minimum cut ratio on a solid grid graph can be calculated in* $\mathcal{O}(|V|^2)$.

Proof. All segments including the cut edges and number of nodes cut off can be calculated in $\mathcal{O}(|V|^2)$ as described in Lemma 7 of [41] which suffices to obtain the minimum cut ratio by the previous theorem. □

Chapter 6

NP-hardness

Before we discuss how to solve the problem (TR-IP) in the one-dimensional and two-dimensional case, we examine the NP-hardness of both cases in order to warrant the applied methods. In Section 6.1 we will give a NP-hardness result in the case of a non-uniform discretization in the one-dimensional case which was proven by a reduction from the knapsack problem in [96]. We will also discuss in which cases the problem can be solved efficiently. As the one-dimensional (TR-IP) is a special case of the two-dimensional (TR-IP) with $M = 1$ the NP-hardness results also apply to the two-dimensional case. In Section 6.2 we will use the minimum bisection problem to conjecture that the problem (TR-IP) is NP-hard even in the case that the set Ξ is binary. The one-dimensional case was discussed in [96] which is an extension of [95]. The two-dimensional results can be found in [74].

6.1 NP-hardness results in one dimension

As shown in the previous chapter in case of a non-uniform discretization the one dimensional (TR-IP) is transformed into the problem (1D-w). If we set $\alpha = 0$, $u \equiv 0$ and $\Xi = \{0, 1\}$, we obtain

$$\begin{aligned} \min_{d \in \mathbb{Z}^N} \quad & \sum_{i=1}^{N} c_i d_i \\ \text{s.t.} \quad & d_i \in \{0,1\} \text{ for all } i \in [N] \quad \text{and} \quad \sum_{i=1}^{N} \sigma_i d_i \leq \Delta \end{aligned}$$

which is the well-known knapsack problem which directly shows that (1D-w) is NP hard. In the case that the discretization is uniform but the set Ξ can be an arbitrary subset of the integers the problem (TR-IP) is also NP-hard as the following result states.

Theorem 6.1 (Proposition 5 in [96]). *The problem* (1D-w) *and the problem* (TR-IP) *without a contiguous* Ξ *are NP-hard.*

Proof. See proof in [96]. □

The proof relies on the construction of a complicated set Ξ while in practice Ξ contains a small number of integers. In Section 8.1 we will see that we can construct a pseudo-polynomial algorithm to solve (1D-w) and the one-dimensional (TR-IP). If we consider the set Ξ to be a fixed input parameter the algorithm is polynomial as Δ can be bounded per Remark 4.2.

6.2 NP-hardness results in two dimensions

We turn to the two-dimensional case and motivate why we conjecture that even the binary (TR-IP) is NP-hard, meaning even with $\Xi = \{0, 1\}$. To this end, we restate the well-studied minimum bisection problem.

The minimum bisection problem (MBP): Given a graph $G = (V, E)$, (MBP) seeks a partition into two sets $S, V \setminus S$ such that $|S|, |V \setminus S| \leq \lceil |V|/2 \rceil$ which minimizes the cardinality of the set of cut edges $C \coloneqq \{\{v_i, v_j\} \in E \mid v_i \in S,\ v_j \in V \setminus S\}$. $|C|$ is called the bisection width.

(MBP) is NP-hard for general graphs and even for unit disc graphs, see [37]. For planar graphs it is conjectured in [80] but not proven that (MBP) remains NP-hard. There exists a polynomial reduction to the minimum bisection problem on subgraphs of the grid $\mathbb{Z} \times \mathbb{Z}$ with an arbitrary number of holes (MBFH), see [80]. So, if (MBFH) is polynomially solvable, so is (MBP) on planar graphs. We now give a polynomial reduction from (MBFH) to the binary (TR-IP) problem. Our reduction is inspired by the ideas in [80].

In order to prove the reduction, we need an auxiliary lemma about the optimal partition of a rectangular grid graph into two sets where one set has size K.

Lemma 6.2 (Lemma 1 in Section 3.2 in [74]). *Let $G = (V, E)$ be a $\tilde{n} \times \tilde{m}$ rectangular subgraph of the infinite grid $\mathbb{Z} \times \mathbb{Z}$. Then for a subset $U \subset G$ of size K it holds that*

$$|\partial U| \geq \min\{\lfloor\sqrt{K}\rceil + \lceil\sqrt{K}\rceil, \tilde{n}, \tilde{m}, \lfloor\sqrt{\tilde{n}\tilde{m} - K}\rceil + \lceil\sqrt{\tilde{n}\tilde{m} - K}\rceil\}$$

where $\partial U = \{\{v_i, v_j\} \in E \mid v_i \in U,\ v_j \in G \setminus U\}$ is the set of cut edges.

Proof. We start with the infinite grid $\mathbb{Z} \times \mathbb{Z}$ before returning to the rectangular subgraph. We want to determine a subset U of the infinite grid with size $K = |U|$ such that the number of edges between U and its complement in the infinite grid U^C is minimized. It is straightforward to see that U is connected. If U contains

two separate connected components, we could shift one component until it becomes adjacent to the other component and reduce the amount of edges between U and its complement in the infinite grid by at least one edge. Thus, U has to be a connected subset.

Let $v_\ell = (x_\ell, y_\ell), v_r = (x_r, y_r), v_b = (x_b, y_b), v_t = (x_t, y_t)$ be the leftmost, rightmost, bottommost, and topmost nodes in the set U. Then U^C contains the node pairs $(x_\ell - 1, y), (x_r + 1, y)$ for $y \in [y_b, y_t]$. If we connect the pairs of the form $(x_\ell - 1, y), (x_r + 1, y)$ by a straight line, we pass through at least one node in U and thus obtain at least two cut edges per pair. The same holds true for the pairs $(x, y_b - 1), (x, y_t + 1)$ for $x \in [x_\ell, x_r]$. Thus, there have to be at least $2(y_t - y_b) + 2(x_r - x_\ell)$ cut edges between U and U^C. For a set of size K this implies that $2\lceil\sqrt{K}\rceil + 2\lfloor\sqrt{K}\rceil$ is a lower bound for the amount of edges between U and U^C (intuitively U should be as close to a square as possible).

We can transfer these insights to the case of the positive orthant, the infinite grid $\mathbb{N} \times \mathbb{N}$. The argumentation from before also holds for the $\mathbb{N} \times \mathbb{N}$ grid but we now assume that for the leftmost node v_ℓ it holds that $v_\ell = (0, y_\ell)$ and for the bottommost node v_b it holds that $v_b = (x_b, 0)$ which implies that the minimum is halved meaning the minimum number of edges between U and its complement in the $\mathbb{N} \times \mathbb{N}$ grid is $y_t + x_r$ or $\lceil\sqrt{K}\rceil + \lfloor\sqrt{K}\rceil$.

We now consider the case of the rectangular subgraph G. If $K < \min\{\tilde{n}, \tilde{m}\}$ the situation coincides with the infinite grid $\mathbb{N} \times \mathbb{N}$. The concavity of the square root ensures that U is connected because in the case of two components the minimum amount of edges would be twice the sum of the square roots of the number of nodes in the components. We obtain the same optimal structures and lower bounds for which examples are shown in the first two grids in Figure 6.1. If $K > |G|/2$ and $\tilde{n}\tilde{m} - K < \min\{\tilde{n}, \tilde{m}\}$, we instead consider U^C and obtain $\lfloor\sqrt{\tilde{n}\tilde{m} - K}\rceil + \lceil\sqrt{\tilde{n}\tilde{m} - K}\rceil$ edges.

It remains the case that $K \geq \min\{\tilde{n}, \tilde{m}\}$ and $\tilde{n}\tilde{m} - K \geq \min\{\tilde{n}, \tilde{m}\}$. Because edges to U^C may arise in only one of the four directions, we need to consider further optimal structures. If $K = c\tilde{n}$ for $c \in \mathbb{N}$, then we can choose U as the first c rows to obtain a set U of size K with exactly $\tilde{n}$ edges to its complement in G, see the third grid in Figure 6.1. The same argumentation holds if $K = c\tilde{m}$ for $c \in \mathbb{N}$. It is evident that this new structure is optimal iff $\tilde{n} < \lceil\sqrt{K}\rceil + \lfloor\sqrt{K}\rceil$ and $\tilde{n} < \lfloor\sqrt{\tilde{n}\tilde{m} - K}\rceil + \lceil\sqrt{\tilde{n}\tilde{m} - K}\rceil$. If $(c+1)\tilde{n} > K > c\tilde{n}$, then either one of the structures derived from the infinite grid $\mathbb{N} \times \mathbb{N}$ is optimal or we can choose U as the first c rows and the part of the next row such that the size matches which implies $\tilde{n} + 1$ cut edges, see the fourth grid in Figure 6.1. Thus, $\tilde{n}$ is also a lower bound in this case. □

Using this lemma we are now able to prove the reduction from minimum bisection below.

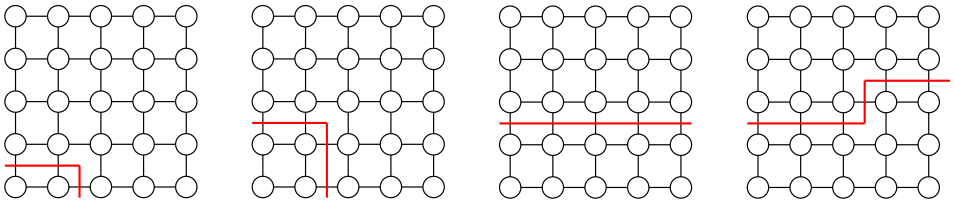

Figure 6.1: Visualization of the optimal cut for different sizes of the set U. From left to right: $K = 2$, $K = 4$, $K = 10$ and $K = 12$. For the first two values of K a shape as close to a square is desirable while for $K = 10$ as an integer multiple of the row size of 5 choosing the first two bottom rows as the set U is optimal. For $K = 12$ we set the first two rows to 1 as well as 2 nodes in the third row. Original figure in [74].

Theorem 6.3 (Theorem 1 in Section 3.2 in [74])**.** *There exists a polynomial reduction from (MBFH) to* (TR-IP) *with a binary control and rational entries in c bounded by* -9 *and* 5 *which become integer values polynomial in the size of the grid graph if multiplied by* n^4*.*

Proof. Let G denote the subgraph of the infinite grid for which we want to obtain a minimum bisection. Without loss of generality, we assume that the subgraph of the grid G has an even number of nodes and more than 2 to be nontrivial. Otherwise, we add a single node not connected to the rest of the graph and the following construction still works. In particular, $n := |G| \geq 4$ is assumed.

We construct a new grid for which an optimal point for (TR-IP) corresponds to a minimal bisection for the bisection problem on the original graph. We replace every node in the graph by a square of n^4 nodes and connect a square with a straight line to another square if the corresponding nodes in G were connected by an edge. The straight line is connected to the middle of the sides of the squares on which sides of the nodes the edge in the original graph was incident to. We choose the node which lies closer to the bottom or the left of the square to decide the ties. The straight lines each contain one node with two incident edges which connect to one square respectively. This ensures that nodes of two different squares do not share an edge. Thus, we obtain a connected graph with n squares of size n^4. In total, we connect the squares by less than $4n$ straight lines each containing one node. We call the set of nodes in a square V_1 and the connecting nodes V_2. We now add nodes and the corresponding edges until we obtain a square grid with a size polynomially bounded in n and in which every node of V_1 has exactly 4 adjacent nodes. We call the newly added nodes the set V_3. An example of the constructed grid is visualized in Figure 6.2. We set $u \equiv 0$ and $\alpha = 1$. For a node v_i in V_1 (nodes in a square) we set the cost $c_{v_i} = -\frac{5}{n^4} - |\{\{v_i, v_j\} \mid v_j \in V_3\}|$ while we set $c_{v_i} = -|\{\{v_i, v_j\} \mid v_j \in V_3\}| = -2$ for $v_i \in V_2$. For v in V_3 we set $c_v = 5$. This construction ensures that $d_v = 0$ for all $v \in V_3$ as the increased objective, specifically $c_v d_v$, outweighs any possible reduction in the jumps regarding this node. Furthermore, the cost of a node $v \in V_2$ is constructed

such that if $d_v = 1$, the term $c_v d_v$ cancels out the jumps to adjacent nodes in V_3. The same holds true for the second term in the definition of the cost term for nodes in V_1. We choose $\Delta = \frac{n^5}{2} + 4n$. This means that half the squares can be set to 1 as well as all the nodes in V_2 which are not part of squares. When we say that we set a node v to 1, we mean that $d_v = 1$.

We now show that it is required for optimality for (TR-IP) for the feasible point to set half of the squares as well as the connecting nodes in V_2 to 1. These feasible points of (TR-IP) then corresponds to feasible points of the minimum bisection and minimizing the cost function of (TR-IP) is equivalent to minimizing the cut lines between the squares which correspond to minimizing the cut edges in the original subgraph. We start by showing that only feasible points of (TR-IP) can be optimal which set the nodes of $\frac{n}{2}$ entire squares to 1.

We first show that it is suboptimal to set more than 0 but less than $n^4 - 25$ nodes in a square to 1. To this end we know that the formula $\min\{\lfloor\sqrt{K}\rfloor + \lceil\sqrt{K}\rceil, n^2, \lfloor\sqrt{n^4 - K}\rfloor + \lceil\sqrt{n^4 - K}\rceil\}$ provides a lower bound for the amount of jumps in the square if we set K nodes to 1. For $5 \leq K \leq n^4 - 25$ the inequality $-\frac{5K}{n^4} + \min\{\lfloor\sqrt{K}\rfloor + \lceil\sqrt{K}\rceil, n^2, \lfloor\sqrt{n^4 - K}\rfloor + \lceil\sqrt{n^4 - K}\rceil\} > 4$ holds which is easy to check due to the concavity of the left side of the inequality. The left side is a lower bound for any feasible point which sets between 5 and $n^4 - 25$ nodes in the square to 1 while the right side is an upper bound for setting all nodes to 0. We remind the reader that the second term $-|\{\{v, w\}|w \in V_3\}|$ in the cost for a node $v \in V_1$ in the square cancels out the jumps to adjacent nodes in V_3. It follows that the cost occurring in the square is higher than a possible reduction in jumps of at most 4 as there are at most 4 nodes from V_2 connected to this square. Because these nodes in V_2 are at least $n^2 - 1$ nodes apart from each other, the same argumentation that the cost in the square outweighs the cost reduction outside the square holds for $0 < K < 5$. We would need one separate component for each jump from a square node to a node in V_2 we want to eliminate, which would instead produce at least two jumps in the square. Thus, a feasible point can only be optimal if for all squares either all the nodes in the square are set to 0 or at least $n^4 - 25$ nodes are set to 1.

We note that it takes a capacity of at least $(n^4 - 25)(\frac{n}{2} + 1) = \frac{n^5}{2} + n^4 - \frac{25n}{2} - 25$ for more than $\frac{n}{2}$ squares to set at least $n^4 - 25$ nodes to 1 which exceeds the capacity bound of $\frac{n^5}{2} + 4n$ for $n \geq 4$. Thus, a feasible point for (TR-IP) can only be optimal if exactly $\frac{n}{2}$ squares are set to 1, because setting fewer squares to 1 would be suboptimal as can be seen by the cost function (setting a whole square to 1 reduces the total cost by at least 1 even if all connecting nodes in V_2 are set to 0).

A feasible point which sets $\frac{n}{2}$ entire squares to 1 minimizes the objective value inside the squares. If also the connecting nodes in V_2 in between these squares are set to 1, the objective is further decreased outside of the squares. It does not effect optimality if a node in V_2 connecting a square which is set to 0 and a square set to 1 is itself set to 0 or to 1 due to the construction of the costs. Setting any other node,

meaning a node in V_1 or V_3, to 1 would increase the objective as previously shown. So we now need to choose the best feasible point from the set of feasible points which adhere to the described conditions in order to obtain an optimal solution for (TR-IP). Thus, the feasible point which sets exactly $\frac{n}{2}$ squares as well as the connecting nodes to 1 is optimal if the amount of straight lines to the remaining squares is minimized, which corresponds to the cut edges for the minimum bisection. The objective value is $-\frac{5n}{2} + C$ where C is the minimal bisection of the original subgraph. Thus, we obtain the desired value by adding $\frac{5n}{2}$ to the objective value of the optimal solution to (TR-IP). □

Remark 6.4. The above proof is the original one from [74] which actually not only works for the minimum bisection problem for grid graphs, meaning node-induced subgraphs of the infinite grid, but for every subgraph of the infinite grid. We can give a more straightforward proof just for grid graphs with the same basic idea.

Alternative proof. Let G denote the grid graph for which we want to obtain a minimum bisection and let n be the number of nodes. We add nodes and edges to the graph until we obtain an $n \times n$ rectangular grid graph in which the original graph is embedded. We denote the set of nodes consisting of all the nodes in the original graph by V_1 and the remaining nodes by V_3. We set $\Delta = \lfloor \frac{n}{2} \rfloor$, $\alpha = 1$ and the previous control $u \equiv 0$. For $v_i \in V_1$ we set $c_{v_i} = -5 - |\{\{v_i, v_j\} \mid v_j \in V_3\}|$. For $v_j \in V_3$ we set $c_{v_j} = 5$.

When we say that we set a node v to 1, we mean that $d_v = 1$. It is straightforward to see that an optimal solution for (TR-IP) will set all nodes in V_3 to 0 as each node has at most 4 adjacent nodes such that any possible reduction in jump costs is always strictly below 5. That means that the cost entry $c_{v_j} = 5$ for the node $v_j \in V_3$ is always larger in absolute value terms. Furthermore, by the construction of the cost terms for nodes in V_1 an optimal solution will set Δ many nodes to 1 as setting a node $v_i \in V_1$ to 1 is always favorable with regards to the objective function. Due to the second term $-|\{\{v_i, v_j\} \mid v_j \in V_3\}|$ in the construction of the cost vector coefficients c, the jumps to adjacent nodes in V_3 are canceled out when setting a node $v_i \in V_1$ to 1. Thus, an optimal point for (TR-IP) is given if jumps to other nodes in V_1 are minimized (which in turn minimizes the number of cut edges for the bisection). The objective value is given by $-5\Delta + C$ where C is the the number of jumps, meaning the number of cut edges for the minimum bisection. □

Corollary 6.5 (Corollary 1 in Section 3.2 in [74])**.** *If (MBFH) is NP-hard and $NP \neq P$, then no (pseudo-)polynomial algorithm for* (TR-IP) *(and by extension for the two-dimensional version of* (1D-w)*) exists.*

Proof. The weights (including $\alpha = 1$) are all integer values polynomial in the size of the original subgraph of the grid if we multiply by n^4 and the trust-region radius is a polynomial in the size of the subgraph of the grid. □

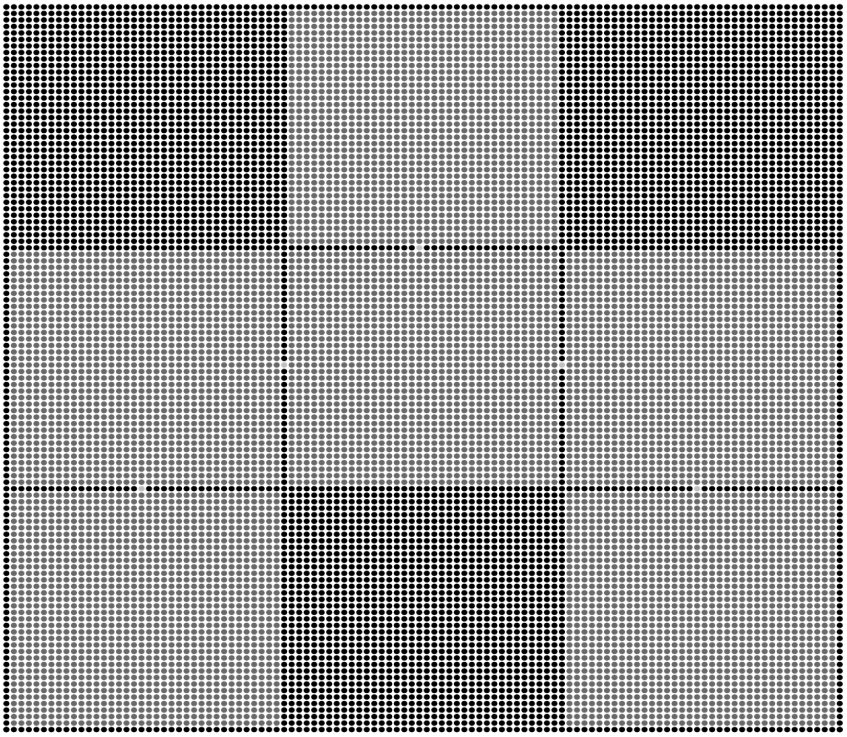

Figure 6.2: The original graph consisted of 6 nodes. Each node becomes one square of 36×36 nodes in V_1. The set V_3 contains the black nodes. The set V_1 is illustrated in dark grey. Two squares are connected by a node in V_2 if the corresponding nodes in the original graph were connected. The set V_2 contains the light grey nodes on the interfaces between these squares of nodes of V_1. Original figure in [74], updated version.

Chapter 7

Structure of the polyhedron P

In this section we will analyze the polyhedron described by the inequalities of (LP) as well as the relationship between the two relaxations (LP) and (LR-Δ). The results were published in [74]. The underlying polyhedron has a special structure which will later lead to valid cutting planes for (IP). To describe this special characteristic, we need the following definition with regards to a feasible point d of (LP).

Definition 7.1 (Definition 1 in Section 4 in [74])**.** We call the set $\Omega \coloneqq \{(i,j)|i,j \in [N]\}$ the **set of all index pairs**. Two index pairs $(i,j),(k,l) \in \Omega$ are **adjacent** if the index pair (k,l) is equal to one of the index pairs $(i+1,j),(i-1,j),(i,j+1),(i,j-1)$. We say that an index pair (k,l) is **adjacent to a subset of index pairs** if it is adjacent to at least one index pair in the subset but is not part of the subset itself.

A subset of index pairs $M \subset \Omega$ is called a **connected component** if it contains only one element or for every index pair $(i,j) \in M$ at least one adjacent index pair is also in M and for any two arbitrary index pairs $(i,j),(k,l) \in M$ the condition $d_{i,j} + u_{i,j} = d_{k,l} + u_{k,l}$ holds. We refer to $d_{i,j} + u_{i,j}$ as the **value of the index pair** (i,j). A connected component is called **maximal** if for every $(i,j) \in M$ the condition $d_{i,j} + u_{i,j} \neq d_{o,p} + u_{o,p}$ for $(o,p) \in \{(i+1,j),(i-1,j),(i,j+1),(i,j-1)\} \cap (\Omega \setminus M)$ holds.

Our goal is to show that all vertices of the polyhedron only take values in Ξ for $u + d$ except in at most one maximal connected component. We will later on in this section use this property to show how to construct a solution to (LR-Δ) from a solution to (LP) and vice versa. In Section 8.3.3 we will construct cuts based on this property of the polyhedron. To prove this result regarding the structure of the polyhedron, we first need to prove an auxiliary lemma.

Lemma 7.2 (Lemma 2 in Section 4 in [74])**.** *For $c_1, c_2 > 0$ and $k, \ell \in \mathbb{R}$ the problem*

$$\begin{aligned} \min_{s_1,s_2} \quad & 0 \\ s.t. \quad & ks_1 + \ell s_2 = 0 \text{ and } -c_1 \leq s_1 \leq c_1 \text{ and } -c_2 \leq s_2 \leq c_2. \end{aligned} \tag{7.1}$$

has a non-trivial optimal solution $s = (s_1, s_2)$ for which $-s$ is also an optimal solution.

Proof. If $k = 0$, then a possible optimal solution is $s = (c_1, 0)$. It also holds that $(-c_1, 0)$ is an optimal solution. The same argument holds for $\ell = 0$ and $s = (0, c_2)$. Let $k \neq 0$ and $\ell \neq 0$, then we can choose $s_1 = -\frac{\ell}{k} s_2$ and $s_2 = \min\{c_2, |\frac{k}{\ell}| c_1\}$ which is feasible for the problem and thus an optimal solution. $-s$ is also an optimal solution because the bounds for s_1, s_2 are symmetric. □

This lemma now allows us to prove the main result of this section.

Theorem 7.3 (Theorem 3 in Section 4 in [74])**.** *Every vertex solution to* (LP) *has at most one maximal connected component with fractional values (values in* $\operatorname{conv}(\Xi) \backslash \Xi$*).*

Proof. Let $(d^*, \delta^*, \beta^*, \gamma^*)$ be optimal for (LP) with two fractional connected components M_1, M_2 and for all $(i, j) \in \Omega$ the equalities $|d^*_{i,j}| = \delta^*_{i,j}$, $|d^*_{i+1,j} + u_{i+1,j} - d^*_{i,j} - u_{i,j}| = \beta^*_{i,j}$ and $|d^*_{i,j+1} + u_{i,j+1} - d^*_{i,j} - u_{i,j}| = \gamma^*_{i,j}$ hold (we can assume this w.l.o.g. because for every feasible point of the relaxation, we can find a feasible point of this form which would have a better objective value in the case that the second or third condition was not met by the original point, see Remark 4.1). Because the set Ω contains a finite number of elements, all connected components have finite size. Thus, we can assume that both connected components are maximal (otherwise we add the missing index pairs to the component).

All index pairs adjacent to one of the connected components have a strictly larger or smaller value. Let q_1 be the value of the index pairs in M_1 and q_2 be the value of the index pairs in M_2. Because Ξ is a finite set, we can find $\underline{b}_1 \coloneqq \max\{\xi \in \Xi | \xi < q_1\}$ and $\bar{b}_1 \coloneqq \min\{\xi \in \Xi | \xi > q_1\}$ for M_1 and $\underline{b}_2 \coloneqq \max\{\xi \in \Xi | \xi < q_2\}$ and $\bar{b}_2 \coloneqq \min\{\xi \in \Xi | \xi > q_2\}$ for M_2.

We now have to distinguish between two cases. For the first case we assume that M_1 and M_2 are adjacent to each other. Because both components are maximal, it follows that $q_1 \neq q_2$. We introduce

$$t_k(i,j) = \begin{cases} s_k & \text{if } (i,j) \in M_k, \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

with $s_k \in [\max\{\underline{b}_k - q_k, q_k - \bar{b}_k, -\frac{1}{2}|q_1 - q_2|\}, \min\{\bar{b}_k - q_k, q_k - \underline{b}_k, \frac{1}{2}|q_1 - q_2|\}]$ with $k \in 1, 2$. We set

$$t \coloneqq t_1 + t_2. \tag{7.3}$$

Because $u_{i,j} \in \Xi$ holds for all $(i, j) \in \Omega$, our choices for the lower bounds $\underline{b}_1, \underline{b}_2$ and for the upper bounds $\bar{b}_1, \bar{b}_2$ ensure that the condition $\operatorname{sign}(d_{i,j} + t(i,j)) = \operatorname{sign}(d_{i,j} - t(i,j)) = \operatorname{sign}(d_{i,j})$ holds, which follows from $\max\{\underline{b}_k - q_k, q_k - \bar{b}_k\} \leq s_k \leq \min\{\bar{b}_k -$

$q_k, q_k - \underline{b}_k\}$ and where we assume that the condition also holds if one side is 0. We obtain that

$$\sum_{(i,j)\in M_1} (|d_{i,j} + t(i,j)| - |d_{i,j}|) + \sum_{(i,j)\in M_2} (|d_{i,j} + t(i,j)| - |d_{i,j}|)$$
$$= \sum_{\substack{(i,j)\in M_1,\\ d_{i,j}>0}} s_1 + \sum_{\substack{(i,j)\in M_1,\\ d_{i,j}<0}} -s_1 + \sum_{\substack{(i,j)\in M_2,\\ d_{i,j}>0}} s_2 + \sum_{\substack{(i,j)\in M_2,\\ d_{i,j}<0}} -s_2$$

and

$$\sum_{(i,j)\in M_1} (|d_{i,j} - t(i,j)| - |d_{i,j}|) + \sum_{(i,j)\in M_2} (|d_{i,j} - t(i,j)| - |d_{i,j}|)$$
$$= \sum_{\substack{(i,j)\in M_1,\\ d_{i,j}>0}} -s_1 + \sum_{\substack{(i,j)\in M_1,\\ d_{i,j}<0}} s_1 + \sum_{\substack{(i,j)\in M_2,\\ d_{i,j}>0}} -s_2 + \sum_{\substack{(i,j)\in M_2,\\ d_{i,j}<0}} s_2$$

hold for any t constructed as above. If the equality

$$\sum_{\substack{(i,j)\in M_1,\\ d_{i,j}>0}} -s_1 + \sum_{\substack{(i,j)\in M_1,\\ d_{i,j}<0}} s_1 + \sum_{\substack{(i,j)\in M_2,\\ d_{i,j}>0}} -s_2 + \sum_{\substack{(i,j)\in M_2,\\ d_{i,j}<0}} s_2 = 0 \tag{7.4}$$

holds, it guarantees that the two points with $\bar{d} = d^* + t$ and $\tilde{d} = d^* - t$ have the same capacity consumption as the point with d^* if we set $|\bar{d}_{i,j}| = \bar{\delta}_{i,j}$ and $|\tilde{d}_{i,j}| = \tilde{\delta}_{i,j}$ for all i, j, which means that they adhere to the capacity constraint of (LP). This also shows that the assumption $\delta^*_{i,j} = |d^*_{i,j}|$ was not a restriction because we could also choose larger $\bar{\delta}_{i,j}$ and $\tilde{\delta}_{i,j}$ if $\delta^*_{i,j}$ were to be larger than $|d^*_{i,j}|$. The conditions (7.2), (7.3) and (7.4) describe the linear program

$$\begin{aligned}
\min_{s_1,s_2} \quad & 0 \\
\text{s.t.} \quad & \sum_{\substack{(i,j)\in M_1,\\ d_{i,j}>0}} s_1 + \sum_{\substack{(i,j)\in M_1,\\ d_{i,j}<0}} -s_1 + \sum_{\substack{(i,j)\in M_2,\\ d_{i,j}>0}} s_2 + \sum_{\substack{(i,j)\in M_2,\\ d_{i,j}<0}} -s_2 = 0, \\
& \max\{\underline{b}_1 - q_1, q_1 - \bar{b}_1, -\tfrac{1}{2}|q_1 - q_2|\} \le s_1 \le \min\{\bar{b}_1 - q_1, q_1 - \underline{b}_1, \tfrac{1}{2}|q_1 - q_2|\}, \\
& \max\{\underline{b}_2 - q_2, q_2 - \bar{b}_2, -\tfrac{1}{2}|q_1 - q_2|\} \le s_2 \le \min\{\bar{b}_2 - q_2, q_2 - \underline{b}_2, \tfrac{1}{2}|q_1 - q_2|\}.
\end{aligned}$$

From our auxiliary lemma we obtain that there exists $s = (s_1, s_2) \neq 0$ for which s and $-s$ solve the linear program. We choose s_1 and s_2 for t accordingly. We set $\bar{\beta}_{i,j} = |\bar{d}_{i+1,j} + u_{i+1,j} - \bar{d}_{i,j} - u_{i,j}|$, $\bar{\gamma}_{i,j} = |\bar{d}_{i,j+1} + u_{i,j+1} - \bar{d}_{i,j} - u_{i,j}|$, $\tilde{\beta}_{i,j} = |\tilde{d}_{i+1,j} +$

$u_{i+1,j} - \tilde{d}_{i,j} - u_{i,j}|$ and $\tilde{\gamma}_{i,j} = |\tilde{d}_{i,j+1} + u_{i,j+1} - \tilde{d}_{i,j} - u_{i,j}|$. We now show

$$\begin{aligned}\operatorname{sign}(\bar{d}_{i+1,j} + u_{i+1,j} - \bar{d}_{i,j} - u_{i,j}) &= \operatorname{sign}(d^*_{i+1,j} + u_{i+1,j} - d^*_{i,j} - u_{i,j}) \\ &= \operatorname{sign}(\tilde{d}_{i+1,j} + u_{i+1,j} - \tilde{d}_{i,j} - u_{i,j})\end{aligned}$$

where we assume that equality also holds if one side is 0. It then follows that $\frac{1}{2}\bar{\beta} + \frac{1}{2}\tilde{\beta} = \beta^*$. If $(i,j),(i+1,j) \in \Omega$ are in the same or in neither fractional component, then it follows from $t(i+1,j) = t(i,j)$ that

$$\begin{aligned}d^*_{i+1,j} + u_{i+1,j} - d^*_{i,j} - u_{i,j} &= d^*_{i+1,j} + t(i+1,j) + u_{i+1,j} - d^*_{i,j} - t(i,j) - u_{i,j} \\ &= d^*_{i+1,j} - t(i+1,j) + u_{i+1,j} - d^*_{i,j} + t(i,j) - u_{i,j}.\end{aligned}$$

If $(i,j) \in M_1$ and $(i+1,j) \in \Omega \setminus (M_1 \cup M_2)$, then $d_{i,j} + t(i,j)$ and $d_{i,j} - t(i,j)$ are bounded by the closest integer values and thus the sign remains the same because $d_{i+1,j} + u_{i+1,j}$ is an integer and $t(i+1,j) = 0$.

If $(i,j) \in M_1$ and $(i+1,j) \in M_2$, then the condition $-\frac{1}{2}|q_1 - q_2| \leq s_k \leq \frac{1}{2}|q_1 - q_2|$ ensures that the sign does not change because $d_{i,j} + u_{i,j} > d_{i+1,j} + u_{i+1,j}$ implies

$$d_{i,j} + t(i,j) + u_{i,j} = q_1 + s_1 \geq q_2 + s_2 = d_{i+1,j} + t(i+1,j) + u_{i+1,j}$$

and vice versa.

Thus, in all possible cases the signs remain the same. By the same argumentation $\frac{1}{2}\bar{\gamma} + \frac{1}{2}\tilde{\gamma} = \gamma^*$ holds. Thus, $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma})$ and $(\tilde{d}, \tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ are both feasible for (LP) and $(d^*, \delta^*, \beta^*, \gamma^*)$ is a convex combination of the two which means that it is not a vertex.

In the second case in which the fractional components are not adjacent, the proof remains the same except that the condition $-\frac{1}{2}|q_1 - q_2| \leq s_k \leq \frac{1}{2}|q_1 - q_2|$ is dropped.

It follows that no feasible point with more than one maximal connected component with fractional values is a vertex of the polyhedron. □

Remark 7.4. We refer to this one maximal connected component with fractional values as the **fractional component**. The proof also works for Ξ which are not contiguous with the same linear programming construction. In this case integer values not in Ξ are also referred to as fractional. The proof also does not require the underlying graph to be a grid and can be adapted to other graph structures by replacing case distinctions of two adjacent nodes in the grid by two adjacent nodes in the new graph structure instead. Later on we will discuss a variation of (TR-IP) for which the underlying graph is fully connected, called (F-TRIP). It follows that the property also holds for the linear programming relaxation for this problem. For a non-uniform discretization, meaning weights in front of δ, this proof should also work with some adaptations as the core idea is that everything is linear if the signs for the absolute value terms do not change (meaning $s_k \in [\max\{\underline{b}_k - q_k, q_k - \bar{b}_k, -\frac{1}{2}|q_1 - q_2|\}, \min\{\bar{b}_k - q_k, q_k - \underline{b}_k, \frac{1}{2}|q_1 - q_2|\}]$ is fulfilled).

Theorem 7.5 (Theorem 4 in Section 4 in [74]). *Every vertex solution* $(d, \delta, \beta, \gamma)$ *of* (LP) *fulfills the capacity constraint with equality or* $u + d \in \Xi^{N\times N}$.

Proof. Let $(d^*, \delta^*, \beta^*, \gamma^*)$ be optimal for (LP) with a fractional connected component for which the capacity constraint is inactive. We define $\underline{b} \coloneqq \max\{\xi \in \Xi | \xi < q\}$ and $\bar{b} \coloneqq \min\{\xi \in \Xi | \xi > q\}$ where q is the fractional value of the connected component M. We define

$$t(i,j) = \begin{cases} s & \text{if } (i,j) \in M, \\ 0 & \text{otherwise} \end{cases}$$

with $s \in [\max(\underline{b} - q, q - \bar{b}), \min(\bar{b} - q, q - \underline{b})]$. Furthermore, we ensure that $\sum_{j=1}^{N}\sum_{i=1}^{N} |d_{i,j} + t(i,j)| \leq \Delta$ and $\sum_{j=1}^{N}\sum_{i=1}^{N} |d_{i,j} - t(i,j)| \leq \Delta$ hold. Because $\sum_{j=1}^{N}\sum_{i=1}^{N} |d^*_{i,j}| < \Delta$, it follows from $\underline{b} - q < 0 < \bar{b} - q$ that we can find a $t \neq 0$ that satisfies all conditions with the same arguments as above. We obtain that $\bar{d} = d^* + t$ and $\tilde{d} = d^* - t$ adhere to the capacity constraint. We set $\bar{\delta}_{i,j} = |\bar{d}_{i,j}|$, $\bar{\beta}_{i,j} = |\bar{d}_{i,j} + u_{i,j} - \bar{d}_{i+1,j} - u_{i+1,j}|$ and $\bar{\gamma}_{i,j} = |\bar{d}_{i,j+1} + u_{i,j+1} - \bar{d}_{i,j} - u_{i,j}|$ and define $\tilde{\delta}$, $\tilde{\beta}$ and $\tilde{\gamma}$ in the same manner. Then $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma})$ and $(\tilde{d}, \tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ are feasible for (LP) which means the convex combination $(d^*, \delta^*, \beta^*, \gamma^*)$ is not a vertex of the polyhedron of (LP). It follows that no feasible point with at least one maximal connected component with fractional values and inactive capacity constraint is a vertex. □

We will use this structure in Section 8.3.3 to obtain valid cuts for our linear relaxation (LP). The previous result also directly implies the connection between the two relaxations (LP) and (LR-Δ).

Corollary 7.6 (Corollary 3 in Section 4 in [74]). *Every vertex solution* $(d, \delta, \beta, \gamma, \mu)$ *of* (LR-Δ) *after the integrality constraint is dropped fulfills* $x + d \in \Xi^{N\times N}$.

Proof. We know that regardless of the integrality constraint every feasible point of (LR-Δ) has a capacity consumption of at most $N^2(\max \Xi - \min \Xi)$. Therefore adding the constraint $\sum_{j=1}^{N}\sum_{i=1}^{N} \delta_{i,j} \leq N^2(\max \Xi - \min \Xi) + 1$ to (LR-Δ) without the integrality constraint does not change the underlying feasible set. It follows from the previous result that every vertex solution fulfills the condition $u+d \in \Xi^{N\times N}$. □

We can, however, make further statements regarding the vertices.

Theorem 7.7 (Theorem 5 in Section 4 in [74]). *Let* $\bar{d}$ *be feasible for the binary* (IP) *with* $\bar{\delta}, \bar{\beta}, \bar{\gamma}$ *constructed as in Remark 4.1. Then* $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma})$ *is a vertex of the polyhedron of* (LP).

Proof. We construct our cost vector c such that $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma})$ is the only optimal point for the corresponding (LP). We set

$$c_{i,j} \coloneqq \begin{cases} -5\alpha & \text{if } u_{i,j} + \bar{d}_{i,j} = 1 \\ 5\alpha & \text{if } u_{i,j} + \bar{d}_{i,j} = 0 \end{cases}$$

and it follows directly that every other choice of d is suboptimal for the corresponding (LP). If a value $d_{i,j}$ is set to a different value than $\bar{d}_{i,j}$, the first term in the objective would increase by 5α, because the $d_{i,j}$ value can only be changed in one direction. This is suboptimal because the second term can not be changed by more than 4α. □

It is evident by the proof that this property does extend to the case in which the β, γ, and δ are weighted by multiplying the constructed c with the maximum weight entry. The proof also works for arbitrary underlying graphs by replacing 5α by α times the sum of 1 and the maximal node degree. This property of the vertices does not translate to the case of a non-binary control even without weights as shown by the following example.

Example 7.8 (Example 1 in Section 4 in [74])**.** Let $\Xi = \{0, 1, 2\}$, $N = M = 3$, $\Delta = 3$, $\alpha = 1$, and $u \equiv 0$. Define

$$d \coloneqq \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ and } d^1 \coloneqq \begin{pmatrix} 0.5 & 0 & 2 \\ 0.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ and } d^2 \coloneqq \begin{pmatrix} 1.5 & 0 & 0 \\ 1.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

We now show that d with the corresponding β, γ, δ is not a vertex of the polyhedron P. We note that d^1 and d^2 are points in the polyhedron P with the corresponding $\beta^1, \gamma^1, \delta^1$ and $\beta^2, \gamma^2, \delta^2$. It is obvious that $d = 0.5d^1 + 0.5d^2$. Due to $u \equiv 0$ it also holds that $(\beta, \gamma, \delta) = 0.5(\beta^1, \gamma^1, \delta^1) + 0.5(\beta^2, \gamma^2, \delta^2)$. Thus $(d, \delta, \beta, \gamma)$ is not a vertex of the polyhedron P. Furthermore, $(d, \delta, \beta, \gamma)$ is optimal for the problem (IP) with the cost vector

$$c = \begin{pmatrix} -2 & 100 & -2.1 \\ -2 & 100 & 100 \\ 100 & 100 & 100 \end{pmatrix}$$

which shows that the optimal solution to (IP) does not have to be a vertex of the polyhedron.

In general the relaxation (LR-Δ) provides a lower bound that is at least as good as the lower bound provided by the relaxation (LP). Both bounds are identical when the underlying polyhedron of the Lagrangian relaxation is already the convex hull of the integer-valued points, see for example p. 125 in [63], which is the case for this problem as we have seen in the previous corollary. We now show that an optimal solution to (LR-Δ) directly gives us an optimal solution to (LP) and vice versa.

Theorem 7.9 (Theorem 6 in Section 4 in [74])**.** *The problems* (LR-Δ) *and* (LP) *are equivalent in the sense that we can construct an optimal solution to one problem from an optimal solution to the other problem.*

Proof. Let $(d^*, \delta^*, \beta^*, \gamma^*)$ be optimal for (LP). We now want to construct a feasible point $(d, \delta, \beta, \gamma, \mu)$ for (LR-Δ) and show the optimality afterwards. We start with

the construction of d. We keep every entry in d^* which is not fractional the same for the corresponding entry in d. If all entries are not fractional, then $(d^*, \delta^*, \beta^*, \gamma^*)$ is already optimal for the integer program and thus also optimal for the Lagrangian relaxation with $\mu^* = 0$ which is a lower bound for the integer program and an upper bound for the linear programming relaxation. All fractional entries are either rounded up or rounded down to the same next value in Ξ depending on which rounding decision decreases the capacity consumption. The values of δ, β, γ are chosen as described in Remark 4.1. The value of μ is set as the quotient of the difference in the objective values regarding the cost function of (IP) and the difference in the capacity consumption of the two points $(d^*, \delta^*, \beta^*, \gamma^*)$ and $(d, \delta, \beta, \gamma)$. The objective value of $(d^*, \delta^*, \beta^*, \gamma^*)$ for the problem (LP) and the objective value of $(d, \delta, \beta, \gamma, \mu)$ regarding the objective function of (LR-Δ) are identical which shows the optimality of $(d, \delta, \beta, \gamma, \mu)$.

Now let $(d^1, \delta^1, \beta^1, \gamma^1, \mu)$ and $(d^2, \delta^2, \beta^2, \gamma^2, \mu)$ with $\sum_{i,j} \delta^1_{i,j} \leq \Delta$ and $\sum_{i,j} \delta^2_{i,j} \geq \Delta$ be optimal for (LR-Δ). In the case that both points are identical, the capacity constraint is fulfilled with equality and $(d^1, \delta^1, \beta^1, \gamma^1)$ is optimal for (IP) and thus for (LP). In the case that both points are not identical the existence is ensured because otherwise we could improve the bound provided by increasing or decreasing μ. We now use the convex combination $(d^3, \delta^3, \beta^3, \gamma^3, \mu)$ of the two points $(d^1, \delta^1, \beta^1, \gamma^1, \mu)$ and $(d^2, \delta^2, \beta^2, \gamma^2, \mu)$ which fulfills the capacity constraint with equality. The convex combination has the same objective value as the other two points regarding the objective function of (LR-Δ). Due to the capacity constraint being fulfilled with equality, the point $(d^3, \delta^3, \beta^3, \gamma^3)$ also has the same objective value regarding the objective function of (LP). Thus, this proves the optimality as the bound provided by the linear programming relaxation is no larger than the bound provided by the Lagrangian relaxation. □

Remark 7.10 (Remark 3 in Section 4 in [74])**.** In particular, the previous proof showed for $\Xi = \{0, 1\}$ and a solution d to the linear programming relaxation (LP) that d^t with

$$u_{i,j} + d^t_{i,j} = \begin{cases} 1 & \text{if } u_{i,j} + d_{i,j} > t, \\ 0 & \text{otherwise} \end{cases}$$

is optimal for every $t \in (0, 1)$ for (LR-Δ). This result parallels the infinite-dimensional thresholding result in [25, Theorem 2.2].

Chapter 8

Algorithmic approaches

In this chapter we discuss the different algorithmic approaches to solve the problems (TR-IP). We begin by discussing the one-dimensional case for which we can use a shortest path approach to obtain a pseudo-polynomial algorithm. Afterwards, we show that we can extend this concept to the case that the underlying graph structure is a tree. We then turn to the two-dimensional case where the underlying structure is a grid. We will use an integer programming solver to tackle these problems and propose a variety of measures to significantly improve the run time. The results for one-dimensional case were originally published in [96] extending [95] in which the ideas were initially presented. The latter does not concern itself with the weighted problem but the original proofs from [95] were straightforward to adjust. Thus, we only give an updated presentation and restate the relevant results from [95] in the updated form from [96] here. A similar approach to the following dynamic programming approach was presented in [75]. The two-dimensional case is based on [74] with multiple major extensions.

8.1 The one-dimensional case

We have seen in Section 5.1 that we can reformulate the problem (TR-IP) in the one-dimensional case as a shortest path problem on a layered DAG with polynomial size. We refer the reader to Section 5.1 for the definitions of the graphs G and G_c. The layered DAG structure directly yields a topological order of the nodes. The topological order implies that the shortest path from s to t can be found in $\mathcal{O}(|V| + |A|) = \mathcal{O}(N \cdot \Delta \cdot |\Xi|^2)$ by iterating over all the nodes, see [30] page 655 ff., which yields the existence of a polynomial algorithm for (TR-IP) (dynamic programming approach). For the one-dimensional case we discuss the more general problem (1D-w) which we have already seen in Section 5.1.1 where we introduced the graph construction and proceed with the following assumption. For (1D-w) the dynamic programming approach is pseudo-polynomial as we can no longer bound Δ by the

size of the underlying grid and thus the size of the constructed graph is pseudo-polynomial. For the graph construction and the following approach we do not require that Ξ is contiguous. We again summarize the setting for the reader.

Setting 8.1 (Setting for (1D-w))**.** Let $c \in \mathbb{R}^N$, $\alpha \in \mathbb{R}_{\geq 0}$, $\Delta \in \mathbb{N}$, $\Xi = \{\xi_1, \ldots, \xi_m\} \subset \mathbb{Z}$ with $\xi_1 < \ldots < \xi_m$ for some $m \in \mathbb{N}$, $u \in \Xi^N$, and $\sigma \in \mathbb{N}^N$ be given.

This dynamic programming approach provides a better worst case complexity than solving the SPP formulation with Dijkstra's algorithm because it avoids the need of a priority queue. It cannot utilize the underlying structure/data to terminate early, however. We set forth to augment Dijkstra's algorithm so that it is able to utilize the problem structure/data and may outperform the dynamic programming approach based on the topological order in practice for larger values of Δ. The derived accelerations transform Dijkstra's algorithm into an A^* algorithm, which arises from Dijkstra's algorithm by adding an additional heuristic function $h : V \to \mathbb{R}$, see [56]. For each node v, the evaluation $h(v)$ estimates the cost to reach the sink t from v. Instead of expanding the node with the lowest path cost, the A^* algorithm expands the node with the lowest sum of the path cost and the heuristic cost estimation. Ties are broken arbitrarily.

While A^* might be a mere heuristic without further assumptions on h, a careful choice of h allows to conserve the optimality of the solution guaranteed by Dijkstra's algorithm. We provide the corresponding concept of a *consistent heuristic* and reference the resulting optimality guarantee below.

Definition 8.2 (see pp. 105-106 in [87])**.** A heuristic function $h : V \to \mathbb{R}$ is called *consistent* if $h(t) = 0$ holds for the sink t and the triangle inequality is satisfied for every edge, specifically

$$w_{v_i,v_j} + h(v_j) - h(v_i) \geq 0 \text{ for all } e = (v_i, v_j) \in A,$$

where w_{v_i,v_j} is the weight for the edge between v_i and v_j.

Theorem 8.3. *Let $G = (V, A)$ be a graph with source s and sink t. The A^* algorithm determines a shortest s-t path if the heuristic is consistent.*

Proof. We refer the reader to [82] page 82 ff. □

If the heuristic function is consistent, the A^* algorithm coincides with Dijkstra's algorithm with the weights $\bar{w}_{v_i,v_j} = w_{v_i,v_j} + h(v_j) - h(v_i)$ and thus solves the SPP formulation in $\mathcal{O}(|V|\log(|V|) + |A|)$, see [63] page 162. This worst case complexity is again higher than the aforementioned topological sorting-based approach described in [30] page 655 ff.

8.1.1 Acceleration of A^* using the Lagrangian relaxation

The reformulation as an RCSPP allows us to use the Lagrangian relaxation of (1D-w) to derive lower and upper bounds for the RCSPP on the costs of the optimal paths in G_c by following the ideas from [36], which are also valid for the SPP on G. Furthermore, we derive a consistent heuristic function for the A^* algorithm from the Lagrangian relaxation. We provide the Lagrangian relaxation with respect to the RCSPP formulation derived in Section 5.1. This Lagrangian relaxation is used to derive a consistent heuristic function for the A^* algorithm that operates on the SPP reformulation of (1D-w). We show how an ε-optimal Lagrange multiplier for the Lagrangian relaxation may be determined in a preprocessing stage in Section 8.1.2.

Lagrangian relaxation

We will discuss how the Lagrangian relaxation can be used to obtain lower and upper bounds and a heuristic function. We follow the general idea from [36] for the bounds in which they consider the RCSPP in the generalized form

$$\mathrm{z}(s,t,\Delta) = \begin{cases} \min\limits_{P} & W(P) \\ \text{s.t.} & P \in \mathbf{P}^{s,t} \text{ and } R(P) \leq \Delta, \end{cases}$$

where $\mathbf{P}^{s,t}$ is the set of all paths from s to t. The cost of a path $P \in \mathbf{P}^{s,t}$ is denoted by $W(P)$, while $R(P)$ is the resource consumption of the path. The corresponding graph in our formulation is G_c, see Section 5.1.2, which we will use in the following explanation.

The authors in [36] propose the idea that in each node of the graph G_c one can solve the shortest path problem to the sink t based on the cost function obtained from the Lagrangian relaxation of the capacity condition with right hand side $\Delta - R_u(v)$ where $R_u(v)$ is the capacity used to reach the current node v in G_c. The problem can be solved for any fixed Lagrange multiplier $\mu \geq 0$. The cost for this shortest path to t with regards to the objective function of the Lagrangian relaxation is a lower bound for the actual constrained shortest path in G_c which has to adhere to the capacity constraint.

In each node the optimal Lagrange multiplier may be different such that we only obtain the tightest bound if we solve the relaxation for this multiplier. One thus would need to solve all the problems individually to get the best possible bounds. Furthermore, one also needs to determine the optimal Lagrange multiplier beforehand or during this process. This would require significant computational demand. So instead we will determine the optimal Lagrange multiplier for our starting node s by a binary search where we solve the all shortest paths problems to t in each node for a given multiplier. We make use of the fact that the dynamic programming approach already calculates the shortest path to t from all nodes in the graph and not just the

shortest path from s without additional computational demand. Thus, we get a set $\mathcal{M}$ of multipliers μ for which we have calculated the shortest paths from every node to t for the given Lagrange multiplier and thus the corresponding lower bounds. We note that if the set of feasible paths for the RCSPP is not empty, there exists a single optimal multiplier or a single interval of optimal multipliers, because the Lagrangian function is concave with respect to μ, see [102].

We can choose the lower bound with the highest value. It is straightforward to see that if we have found a path from s to t which adheres to the capacity constraint, we obtain an upper bound for the cost of an optimal constrained shortest path. An upper bound inducing path is also given if the shortest path from a node to t calculated as part of the Lagrangian relaxation already fulfills the capacity condition with regards to the remaining capacity $\Delta - R_u(v)$ (because the total path is then feasible for the RCSPP). If the sum of the cost to reach a node and the lower bound calculated by the presented approach is higher than this upper bound, the path can not be optimal which implies that this path can be cut off (this holds for upper bounds in general). For more details regarding the bound construction, we refer the reader to [36]. Additionally, the calculated lower bounds for the costs of the remaining paths lead to the construction of the following heuristic function.

Heuristic function

The Lagrangian relaxations can also be used to obtain a consistent heuristic function for the A^* algorithm. By construction of G_c, an equivalence class of vertices of G is a node in G_c. This allows us to extend the Lagrangian relaxations for the nodes of G_c to the nodes of G and thus construct a heuristic function with regards to G. We recall that a node $v_j \in V$ satisfies $v_j \in [v_i]$ for $[v_i] \in V_c$ if

$$\tilde{d}(v_j) = \tilde{d}(v_i) \quad \text{and} \quad \ell(v_j) = \ell(v_i).$$

Theorem 8.4 (Theorem 12 in [96])**.** *Let Setting 8.1 hold. Then for all $\mu \geq 0$ the heuristic function*

$$h_\mu(v) = \mathrm{LR}([v], t, \tilde{r}(v), \mu) = -\mu\tilde{r}(v) + \zeta([v], t, \mu)$$

is consistent for the A^* algorithm on G where $\zeta([v], t, \mu)$ denotes the cost of the shortest path from v to t in G_c for the multiplier μ.

Proof. See proof in [96]. □

For each value of μ we obtain valid lower bounds. Thus, in each node we want to choose the tightest, meaning highest, lower bound. This is realized in the following heuristic function which always chooses this highest lower bound in each node.

Theorem 8.5 (Proposition 13 in [96]). *Let Setting 8.1 hold. Let $\mathcal{M} \subset [0, \infty)$ with $|\mathcal{M}| < \infty$ be given. Then the heuristic function*

$$h_{\mathrm{LR}}(v) = \max_{\mu \in \mathcal{M}} \mathrm{LR}([v], t, \tilde{r}(v), \mu)$$

is consistent on G.

Proof. See proof in [96]. □

8.1.2 Preprocessing stage

We show that the calculation of the shortest paths in the relaxed graphs (in the form of G_c) and the determination of an ε-optimal Lagrange multiplier may take place in a preprocessing stage. To this end, we use the equivalence of feasible and optimal points for (1D-w) and feasible and shortest paths for the corresponding SPPs and RCSPPs as is argued in Theorems 5.1 and 5.2. We use a binary search with the initial upper bound

$$\bar{b} \coloneqq c_{\max} + 2\alpha \quad \text{with} \quad c_{\max} \coloneqq \max_{i \in [N]} |c_i|$$

and the initial lower bound $\underline{b} = 0$, which is given in Algorithm 2. We will see in Lemma 8.7 that the upper bound is choosen suffiently large. In each iteration of

Algorithm 2 Binary Search

Input: $G_c = (V_c, A_c)$, source s, sink t, weights $w : A_c \to \mathbb{R}_{\geq 0}$, $\bar{b} = c_{max} + 2\alpha$, $\underline{b} = 0$, $\varepsilon > 0$

Output: all shortest paths to t for all calculated values of μ

while $\bar{b} - \underline{b} \geq \varepsilon$ **do**
  $\mu \leftarrow \frac{\bar{b}+\underline{b}}{2}$
  $\tilde{w}(v_i, v_j) \leftarrow w(v_i, v_j) + \mu\sigma_{\ell(v_j)}|\tilde{d}(v_j)|$ for all edges $e = (v_i, v_j)$
  $P_\mu \leftarrow$ calculate shortest paths from all nodes v to t for G_c with weights $\tilde{w}$
  $p_\mu \leftarrow$ shortest s-t path in P_μ that minimizes the resource consumption
  if resource consumption of p_μ from s to t exceeds Δ **then**
    $\underline{b} \leftarrow \mu$
  else
    if resource consumption of p_μ equals Δ **then**
      Terminate early, p_μ is optimal for G.
    end if
    $\bar{b} \leftarrow \mu$
  end if
end while
return shortest paths to t and the corresponding μ

Algorithm 2, an *all shortest paths problem* to t is solved. Due to the LDAG structure,

the latter can be solved with the algorithm detailed in [30] page 655 ff (dynamic programming approach). If several paths have the same cost, the path with the lower resource consumption is chosen. Remaining ties are broken arbitrarily. Because a duality gap may prevail, none of the calculated shortest s-t paths are necessarily optimal for the (SPP) on G. If, however, the remaining capacity for one of the calculated shortest paths is 0, the path is already optimal for G, which we state in Theorem 8.6 below. Afterwards, we see in Theorem 8.9 that the binary search is well-defined, meaning that it terminates after finitely many steps, and returns a Lagrange multiplier that is within an ε-distance of an optimal one.

Because the remaining capacity of a path is given by the difference between the sum of $\sigma_{\ell(v)}|\tilde{d}(v)|$ over all nodes v in the path and the input Δ, the Lagrangian relaxation of the resource constraint can be integrated into the weight function by adding the term $\mu\sigma_{\ell(v)}|\tilde{d}(v)|$ when an edge that points to the node v is added to the graph. This is done in Algorithm 2 line 3 and gives a one-to-one relation to the Lagrangian relaxation term $\mu\left(\sum_{i=1}^{N}\sigma_i|d_i| - \Delta\right)$ in terms of the problem formulation (1D-w).

Theorem 8.6 (Proposition 14 in [96])**.** *Let Setting 8.1 hold. Let d^* be optimal for the relaxed problem*

$$(\dagger_\mu) \quad \begin{aligned} \min_d \ & \sum_{i=1}^{N} c_i d_i + \alpha \sum_{i=1}^{N-1} |u_{i+1} + d_{i+1} - u_i - d_i| + \mu\left(\sum_{i=1}^{N}\sigma_i|d_i| - \Delta\right) =: C_\mu(d) \\ \text{s.t. } & u_i + d_i \in \Xi \text{ for all } i \in [N] \end{aligned}$$

for a fixed $\mu \geq 0$. If $\sum_{i=1}^{N}\sigma_i|d_i^*| - \Delta = 0$, then d^* is optimal for (1D-w)*.*

Proof. See Online Supplement to [96]. □

If a vector d as in the claim of Theorem 8.6 (or an optimal path for the RCSPP reformulation) is found, then the binary search may terminate early and the A^* algorithm or any other solution algorithm may be skipped entirely. In order to state that the binary search terminates after finitely many steps and returns an ε-optimal Lagrange multiplier, we need two auxiliary lemmas which are provided below.

Lemma 8.7 (Lemma 15 in [96])**.** *If Setting 8.1 holds and $\mu \geq c_{\max} + 2\alpha$, $d \equiv 0$ minimizes $(\dagger_\mu)$.*

Proof. See Online Supplement to [96]. □

Lemma 8.8 (Lemma 16 in [96])**.** *Let Setting 8.1 hold. Let $\mu \geq 0$ be fixed. Let d^* be a minimizer of $d \mapsto \sum_{i=1}^{N}\sigma_i|d_i|$ over the set of optimal solutions of $(\dagger_\mu)$. Let μ^* be*

optimal for

$$\arg\max_{\hat{\mu}\geq 0} \min_{d} \sum_{i=1}^{N} c_i d_i + \alpha \sum_{i=1}^{N-1} |u_{i+1} + d_{i+1} - u_i - d_i| + \hat{\mu} \left(\sum_{i=1}^{N} \sigma_i |d_i| - \Delta \right) \tag{8.1}$$
$$\text{s.t. } u_i + d_i \in \Xi \text{ for all } i \in [N].$$

(i) If $\sum_{i=1}^{N} \sigma_i |d_i^*| - \Delta > 0$ *holds, then the inequality* $\mu < \mu^*$ *holds.*

(ii) If $\sum_{i=1}^{N} \sigma_i |d_i^*| - \Delta < 0$ *holds, then the inequality* $\mu \geq \mu^*$ *holds.*

Proof. See Online Supplement to [96]. □

Theorem 8.9 (Proposition 17 in [96])**.** *Let Setting 8.1 hold. Let* $G_c = (V_c, A_c)$ *be constructed as described in Section 5.1. Then Algorithm 2 terminates after finitely many iterations. Moreover, the returned value of* μ *differs at most by* ε *from an optimal Lagrange multiplier.*

Proof. See [96]. □

8.2 Extension to trees

In this section we discuss how to use dynamic programming in a similar way to the presented shortest path approach to solve the problem (TR-IP) if the underlying graph structure is a tree. We first introduce the algorithm for trees for which each node has at most two children, see Algorithm 3. Afterwards we extend the algorithm to the problem for arbitrary trees.

In the one-dimensional case each node of the graph was replaced by a layer for the shortest path approach with a size of $|\Xi| \times (\Delta + 1)$. This encoded the choice for the control and the remaining capacity and the shortest path approach assigned a cost for each entry. The same construction is applied for the tree and is fulfilled by the array M in Algorithm 3, which is initialized with all values being ∞.

The tree has levels based on the depth (distance to the root node). We start at the highest level, meaning furthest from the root node, and then continue with each subsequent lower level until we reach the root node, represented by the for-loop starting in line 2. We check each node $\bar{n}$ in the level in an arbitrary order, see line 3. In each node we have to consider each combination of the control values Ξ with a remaining capacity r. We accomplish this as we iterate over all control values in Ξ and distinguish between 3 cases depending on the number of children.

Case 1: If the node $\bar{n}$ is a leaf, meaning that it has no children, we do not have to consider any previous choice. Thus, the remaining capacity is uniquely determined by each choice of the control value $\bar{\xi}$ and is given as $\Delta - |d|$ where d is the change

of the control value compared to the initial control value $u_{\bar{n}}$, see line 5. As we do not consider any previous decisions, the cost is given by $c_{\bar{n}}d$ for the corresponding entries in M. The other entries are not changed as the encoded combinations can not be attained and the values remain as ∞.

Case 2: If the node $\bar{n}$ has one child C_1, each combination of the control value and remaining capacity r up to the bound of $\Delta - |d|$ is considered. We now have to consider the previous choices encoded in the child node. The cost given in the entry $M(\bar{n}, \bar{\xi}, r)$ not only consists of the term $c_{\bar{n}}d$ but additionally consists of the minimal combination over all $\xi_i \in \Xi$ of an entry $M(C_1, \xi_i, r + |d|)$ for the correct remaining capacity of $r + |d|$ in the array regarding the child C_1 and the cost of a possible jump $|\bar{\xi} - \xi_i|$ (difference in control values), see line 10.

Case 3: If the node $\bar{n}$ has two children C_1 and C_2, we have to consider any possible combination of capacity consumption in these two nodes, which are denoted by $cap1$ and $cap2$. The corresponding remaining capacity in the node is $r = \Delta - |d| - cap1 - cap2$. As we iterate over the possible values for the remaining capacity, see line 13, we ensure that the sum of $cap1$ and $cap2$ is always $\Delta - |d| - r$. This is accomplished by iteratively setting the value of $cap1$ from 0 to $\Delta - |d| - r$ while $cap2$ is set such that $cap1 + cap2 = \Delta - |d| - r$ holds, see line 14 and 15. Thus, the remaining capacities in the children nodes have to be given by $\Delta - cap1$ and $\Delta - cap2$ respectively. We now consider the minimal cost combinations, previously described for one child, for both children and minimize regarding the possible combinations of $cap1$ and $cap2$. This means that the cost now includes the term $c_{\bar{n}}d$, the minimal combination for all $\xi_i \in \Xi$ of an entry $M(C_1, \xi_i, \Delta - cap1)$ and the minimal combination for all $\xi_j \in \Xi$ of an entry $M(C_2, \xi_j, \Delta - cap2)$ for fixed $cap1$ and $cap2$, see line 16. The final entry is the lowest cost calculated for all possible value combinations for $cap1$ and $cap2$.

We do not have to consider previous information except for the information given in the children which is the same principle used in the one-dimensional case. All operations can be performed in polynomial time which allows for an efficient algorithm for underlying tree graphs with at most two children in each node.

We now discuss how to extend this algorithm to arbitrary trees. For nodes with at most 2 children we run the Algorithm 3 as before. For nodes with more children we employ a slightly different procedure in which we replace the node and its children by a binary tree. Let n be a node in the original tree with k children for an arbitrary $k \in \mathbb{N}$. Let ℓ be the smallest integer such that $k \leq 2^\ell \leq 2k$. We build a binary tree structure such that n is the root and the k children are the leafs. The remaining nodes are copies of n. The tree has depth ℓ. We do the same procedure for this tree as described in Algorithm 3, starting in the nodes with only leaves as children with two slight variations. The remaining capacity is only reduced by the choice of d in

the root node. Furthermore, the linear cost term $c_{\bar{n}}d$ is dropped except in the root node. We note that it is suboptimal if the attained value in a child which is a copy of n is different from the value taken in the parent node as this only adds additional jumps with costs of the form $|\bar{\xi} - \xi_i|$.

This results in an algorithm which solves the problem for arbitrary trees. As the number of nodes in the constructed binary trees are, in the worst case, less than double the number of children of the node from the original tree, the algorithm is still polynomial (pseudo-polynomial in the case of integer weights).

Algorithm 3 Dynamic programming for binary trees (max two children per node)

Input: Tree with n nodes and root node n_r, cost vector c, set of control values Ξ, previous control u, trust-region radius Δ .

Initialize array M of size $n \times |\Xi| \times (\Delta + 1)$ with values ∞
for level ℓ in the tree (starting in the highest) **do**
  for node $\bar{n}$ in level ℓ **do**
    for control $\bar{\xi}$ in Ξ **do**
      $d = \bar{\xi} - u_{\bar{n}}$
      if set of children $C(\bar{n}) = \emptyset$ **then**
        $M(\bar{n}, \bar{\xi}, \Delta - |d|) = c_{\bar{n}}d$
      else if set of children $C(\bar{n}) = \{C_1\}$ **then**
        for $r \in \{0, \ldots, \Delta - |d|\}$ **do**
          $M(\bar{n}, \bar{\xi}, r) = c_{\bar{n}}d + \min_{\xi_i \in \Xi}(M(C_1, \xi_i, r + |d|) + |\bar{\xi} - \xi_i|)$
        end for
      else if set of children $C(\bar{n}) = \{C_1, C_2\}$ **then**
        for $r \in \{0, \ldots, \Delta - |d|\}$ **do**
          for $cap1 \in \{0, \ldots, \Delta - |d| - r\}$ **do**
            $cap2 = \Delta - |d| - r - cap1$
            $M(\bar{n}, \bar{\xi}, r) = \min\{M(\bar{n}, \bar{\xi}, r), c_{\bar{n}}d + \min_{\xi_i \in \Xi}(M(C_1, \xi_i, \Delta - cap1) + |\bar{\xi} - \xi_i|) + \min_{\xi_j \in \Xi}(M(C_2, \xi_j, \Delta - cap2) + |\bar{\xi} - \xi_j|)\}$
          end for
        end for
      end if
    end for
  end for
end for
Return: $\min_{r \in \{0,\ldots,\Delta\}} \min_{\xi_i \in \Xi} M(n_r, \xi_i, r)$ and arguments

8.3 The two-dimensional case

In the two-dimensional case we have conjectured that the problem (TR-IP) is NP-hard and we have discussed that the best known algorithm for the related minimum bisection problem on a solid grid graph has a time complexity of $\mathcal{O}(|V|^4)$, see [41] for

the algorithm. Even if the binary (TR-IP) is not NP-hard, we believe it is likely that a polynomial algorithm would also have a high run time complexity. Due to these reasons we propose to employ an integer programming solver for solving (TR-IP). We derive several tools to reduce the run time of the integer programming solver. Specifically, we propose a primal heuristic, a branching priority and cutting planes.

8.3.1 Primal heuristic

Substructures of the grid can correspond to one-dimensional problems of (TR-IP) as we have seen when we discussed the relationship to graph-based problems and that the shortest path approach does not extend to the two-dimensional case, see Chapter 5. We want to use this property to improve any feasible point found by the integer programming solver. Let $(\tilde{d}, \tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ be a feasible point of (IP) with $N = M$ for sake of clarity (but the idea also works for $N \neq M$). We assume N to be even for sake of clarity but the arguments also hold for odd N. We consider the reduced problem

$$
\text{(red IP)} \quad \begin{aligned} \min_{d,\delta,\beta,\gamma} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} c_{i,j} d_{i,j} + \alpha \left(\sum_{i=1}^{N-1}\sum_{j=1}^{N} \beta_{i,j} + \sum_{i=1}^{N}\sum_{j=1}^{N-1} \gamma_{i,j} \right) \\ \text{s.t.} \quad & (d, \delta, \beta, \gamma) \in P_\Delta, \\ & d \in \mathbb{Z}^{N \times N} \text{ and } d_{i,j} = \tilde{d}_{i,j} \text{ for } i \in \{2, 4, \dots, N\}, j \in \{1, \dots, N\}. \end{aligned}
$$

Like the one-dimensional problem, the problem (red IP) can be solved by a shortest path approach with the same graph structure and a slight variation of the weights to consider the jumps to the fixed nodes.

The graph construction is similar to the one in Chapter 5 as we determine the control values of the nodes starting in the first row and continuing through the rows not already fixed but with two changes. The first change is that we need to consider the jumps to already fixed nodes given by $\tilde{d}_{i,j}$ which we do by adjusting the weights of the edges. The second change is that we need to adjust the weights as we start fixing a new row to model that we do not consider jumps from the last node of the previous row to the first node of the current row.

We construct a graph $G = (V, A)$ with $\frac{N^2}{2}|\Xi|\Delta + 2$ nodes including the source and the sink. There are $\frac{N^2}{2}$ layers with respectively $|\Xi|\Delta$ nodes where the nodes of the first layer are connected to the source and the nodes of the last layer are connected to the sink. Each node is only connected to nodes in the previous and following layer. We describe a node $v \in V$, excluding the source and the sink, as a triplet $v = (j, \delta, \eta) \in [\frac{N^2}{2}] \times \{x - y | x, y \in \Xi\} \times \{0, \dots, \Delta\}$. We define the notation $\ell(v) = j$, $p(v) = \delta$ and $\tilde{r}(v) = \eta$. (We note that $p(v)$ had the role of $\tilde{d}(v)$ as defined in Section 5.1.)

An edge e between two nodes $v_k, v_l \in V \setminus \{s,t\}$ is defined by

$$e = (v_k, v_l) \in A \quad :\Longleftrightarrow \quad \begin{cases} \ell(v_l) = \ell(v_k) + 1, \\ \text{there exists } (a,b) \in A \text{ with } b = v_k, \\ \tilde{r}(v_l) = \tilde{r}(v_k) - |p(v_l)|. \end{cases}$$

The first condition enforces the layer structure, while the second and last conditions ensure that the capacity constraint holds inductively. For a clearer presentation, we introduce $i(v) = a$ and $j(v) = b$ where $\ell(v) = (2a-1)N + b$. The weight of an edge $e = (v_k, v_l) \in A$ with $v_k, v_l \in V \setminus \{s,t\}$ is given by

$$\begin{aligned} w_{(v_k,v_l)} =& c_{i(v_l),j(v_l)} + \alpha \begin{cases} |u_{i(v_l),j(v_l)} + p(v_l) - u_{i(v_k),j(v_k)} - p(v_k)| & \text{if } i(v_k) = i(v_l), \\ 0 & \text{otherwise,} \end{cases} \\ &+ \alpha \begin{cases} |u_{i(v_l),j(v_l)} + p(v_l) - u_{i(v_l)+N,j(v_l)} - \tilde{d}_{i(v_l)+N,j(v_l)}| & \text{if } i(v_l) < N, \\ 0 & \text{otherwise,} \end{cases} \\ &+ \alpha \begin{cases} |u_{i(v_l),j(v_l)} + p(v_l) - u_{i(v_l)-N,j(v_l)} - \tilde{d}_{i(v_l)-N,j(v_l)}| & \text{if } i(v_l) > 1, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The first case distinction deals with jumps between nodes in the same layer. The second case distinction models the jumps to nodes in the same column but following fixed row. The final case distinction models the jumps to nodes in the same column but in the preceding fixed row. These case distinctions can be seen in Figure 8.1 in which the edges are denoted by A, B, C accordingly. We note that the second case distinction is not needed because N was assumed to be even, thus the first subcase is always fulfilled, but the distinction is done anyway for sake of completeness for the case of an odd N. The source $s = (0, \emptyset, \Delta)$ is connected to all $v_k \in V$ in the first layer with sufficient remaining capacity, that is

$$(s, v_k) \in A \quad :\Longleftrightarrow \quad \ell(v_k) = 1 \text{ and } \tilde{r}(v_k) = \Delta - |p(v_k)|.$$

The weight is given by $w_{(s,v_k)} = c_1 p_N(v_k) + |u_{2,1} + \tilde{d}_{2,1} - u_{1,1} - p(v_k)|$. The sink $t = (NM+1, \emptyset, 0)$ is connected to each node $v_l \in V$ in the last layer that has an incoming edge, that is

$$(v_l, t) \in A \quad :\Longleftrightarrow \quad \text{there exists } v_k \in V \text{ such that } (v_k, v_l) \in A.$$

The weights have the value zero, that is $w_{(v_l,t)} = 0$. Just like the shortest path approach for the one-dimensional case, we can obtain an optimal point for (red IP) by solving the shortest path problem from s to t.

We now show that a solution of (red IP) will always have an objective value no worse than the point $(\tilde{d}, \tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ used for the construction of the problem.

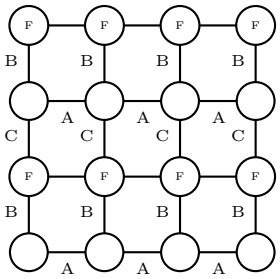

Figure 8.1: The nodes marked as F are the fixed nodes in the grid. Each edge ending in at least one node not fixed beforehand represents one of the case distinction in the weights definition for the shortest path approach, specifically A stands for the first, B for the second and C for the last case. Original figure in [74].

Theorem 8.10 (Theorem 10 in Section 5.2 in [74])**.** *Let* $(\tilde{d}, \tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ *be a feasible point of* (IP) *with* $N = M$*. Then every solution of the problem* (red IP) *is feasible for* (IP) *and has a lower or equal objective value compared to the objective value of* $(\tilde{d}, \tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ *regarding the objective function of* (IP)*.*

Proof. The problem (red IP) can be derived from (IP) by adding the constraints $d_{i,j} = \tilde{d}_{i,j}$ for $i \in \{2, 4, \ldots, N\}, j \in \{1, \ldots, N\}$. Thus, the objective functions coincide and the feasible set of (red IP) is a subset of (IP). The point $(\tilde{d}, \tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ is feasible for (red IP). Thus, any solution of (red IP) has a objective value lower or equal to the objective value of $(\tilde{d}, \tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ and is feasible for (IP). □

We can now use this improved feasible point to construct a new problem of the form (red IP) with different fixed d entries and repeat this process. In our algorithm we alternately fix the entries $d_{i,j}$ such that we alternate between fixing the odd rows, the even rows, the odd columns and the even columns. We continue this process until none of the four variations produce an improved feasible point. This process terminates finitely as there are only finitely many feasible points for the problem (IP) and in each finite loop we improve the objective value.

8.3.2 Branching priority

We propose a branching priority motivated by the primal heuristic. In the previous subsection we observed that fixing half of the nodes allows us to obtain a one-dimensional problem and use a shortest path approach to obtain an optimal solution. This shows us that fixing the nodes in an order which follows this strategy might be preferable in order to come closer to a problem class that is more efficiently solvable as seen in the previous subsection. Thus, we raise the priority of a node if it is in an even row and/or column first as these nodes are the most significant to progress towards a preferable case for the primal heuristic.

8.3.3 Cutting planes

We introduce two different classes of cutting planes which use the structure of the polyhedron presented in Chapter 7, namely that we have one single fractional component. While the original constraints describing the polyhedron are very sparse except for the capacity constraint, the resulting cutting planes will not be sparse but contain a number of variables in the order of the size of the fractional component. As the latter may be as large as the whole graph the resulting inequalities would in turn be very dense. Thus, in computational practice, we add the cutting planes conservatively to ensure that the improvement of the linear programming relaxation is more impactful on the run time than the increase in computational demand for the relaxation. While we will assume $\Xi = \{0, 1\}$ for the derivation of the cutting planes, we briefly discuss how this can be extended to the more general case afterwards.

Cutting plane derived from a fully connected graph

As shown in Chapter 7 any vertex solution to (LP) has one maximal connected component with the same values $u+d$ called the fractional component. Compared to any feasible Ξ-valued point with the same capacity consumption on this component, the relaxation does not create any jumps inside of the component while feasible Ξ-valued points do. To penalize this behaviour of the relaxation solution, we can construct a cutting plane which enforces that the capacity consumption on the fractional component is reflected in the amount of jumps.

The notation in this as well as for the following cutting plane is as follows. The tuple $(d^*, \delta^*, \beta^*, \gamma^*)$ denotes an arbitrary but fixed optimal solution to (LP). The set of index pairs $F = \{(i,j) \in \Omega \mid u_{i,j} + d^*_{i,j} \notin \Xi\}$ contains those for which $u + d^*$ has fractional values. The set of index pairs $H = \{(i,j) \in \Omega \mid u_{i,j} + d^*_{i,j} \in \Xi,\ d^*_{i,j} \neq 0\}$ contains those for which $u + d^*$ has values in Ξ and a strictly positive capacity consumption (the corresponding entries of d^* are non-zero). For the capacity bound Δ and the capacity consumption outside of the fractional component $\Delta_{out} \coloneqq \sum_{(i,j)\in\Omega\setminus F} \delta^*_{i,j} = \sum_{(i,j)\in H} \delta^*_{i,j}$, we obtain that $\Delta_r \coloneqq \Delta - \Delta_{out}$ is the implied capacity bound for the fractional component. Due to Theorem 7.5, this coincides with the capacity consumption of d^* on F. We denote the largest connected component in F in which the previous control values $u_{i,j}$ are identical by G. The component G can be interpreted as a connected subgraph of the grid and we use $\mathcal{G} = (V, E)$ to denote the corresponding graph.

Remark 8.11. In the following we use a slightly changed definition for the minimum cut ratio compared to the one given in Section 5.4, which uses the bound $\frac{|V|}{2}$ where V is the set of nodes of the graph $\mathcal{G}$. Instead we now use the bound Δ_r. When we

use the term minimum cut ratio in the following we thus refer to the term

$$\rho \coloneqq \min_{U \subset V,\ 0<|U|\leq \Delta_r} \frac{|\partial U|}{|U|}.$$

From the definition of ρ, it follows that $\rho|U| \leq |\partial U|$ holds for all possible subsets $U \subset V$ with $0 < |U| \leq \Delta_r$. Consequently, every feasible integer assignment on G with capacity consumption $k \in \mathbb{N}$ less than or equal to Δ_r corresponds to such a subset U with $|U| = k$. The cut size $|\partial U|$ is the amount of jumps for this integer assignment within G. Consequently, multiplying ρ with the capacity consumption of a feasible integer assignment that consumes less than or equal to Δ_r capacity on G gives a lower bound on its amount of jumps.

Theorem 8.12 (Theorem 7 in Section 5.1 in [74]). *Let $\Xi = \{0, 1\}$. Let Δ_{out} be a fixed integer value between 0 and Δ. Let G, H be disjoint sets of index pairs, $|H| = \Delta_{out}$ and G is a connected component with the same value $u_{i,j}$ for every $(i, j) \in G$. Let $\mathcal{G}(V, E)$ be the graph corresponding to G with a minimum cut ratio ρ. Every feasible point $(d, \delta, \beta, \gamma)$ of* (IP) *which fulfills $\sum_{(i,j)\in H} \delta_{i,j} = \Delta_{out}$ also fulfills the inequality*

$$0 \leq -\rho \sum_{(i,j)\in G} \delta_{i,j} + \sum_{\substack{((i,j),(i+1,j))\\ \in G\times G}} \beta_{i,j} + \sum_{\substack{((i,j),(i,j+1))\\ \in G\times G}} \gamma_{i,j}.$$

For an M sufficiently large the inequality

$$0 \leq -\rho \sum_{(i,j)\in G} \delta_{i,j} + \sum_{\substack{((i,j),(i+1,j))\\ \in G\times G}} \beta_{i,j} + \sum_{\substack{((i,j),(i,j+1))\\ \in G\times G}} \gamma_{i,j} + M(\Delta_{out} - \sum_{(i,j)\in H} \delta_{i,j})$$

holds for every feasible point of (IP).

Proof. The first part follows from the construction of ρ and the insights presented above. The terms $\sum \beta_{i,j} + \sum \gamma_{i,j}$ can be interpreted as the cut edges for a cut corresponding to the integer assignment. As the product of the used capacity on G (given in the form $\sum_{(i,j)\in G} \delta_{i,j}$) and the minimum cut ratio is a lower bound for the amount of cut edges, we obtain that the first inequality holds. The second part is the so-called big-M formulation of the implication. We see that the second inequality coincides with the first inequality in the case that $\Delta_{out} = \sum_{(i,j)\in H} \delta_{i,j}$. It is straightforward to see that a sufficiently large choice for M is $M = \rho|G|$. (For the actual implementation one would obviously want to choose a smaller M.) □

We again state that for the setting of the fractional component it always holds that $\Delta_{out} = \sum_{(i,j)\in H} \delta_{i,j}$ by construction of the set H which will later ensure that the linear programming relaxation is cut off by the cutting plane. We do not expect that we can calculate ρ for arbitrary subgraphs of the grid without significant

computational demand as the subgraphs might even contain holes. Instead, we want to use a simpler structure, a fully connected graph, for which we can determine the minimum cut ratio $\tilde{\rho}$ of the resulting subgraph in a straightforward manner. We will study a related problem which is our problem (TR-IP) with a different underlying graph structure where we have replaced the grid with a fully connected graph. We will see that this problem can be solved easily and derive cutting planes based on the minimum cut ratio for this problem. Afterwards we transfer the cutting planes to our original (TR-IP). To this end we introduce the problem

$$
\begin{aligned}
\text{(F-TRIP)} \qquad \min_{d,\zeta} \quad & c^T d + \sum_i \sum_{j>i} \zeta_{i,j} \\
\text{s.t.} \quad & -\zeta_{i,j} \leq u_i + d_i - u_j - d_j \leq \zeta_{i,j} \quad \forall i,j \in \{1,\dots,\tilde{N}\}, j > i, \\
& \sum_i |d_i| \leq \Delta, \\
& d_i + u_i \in \{0,1\} \quad \forall i \in \{1,\dots,\tilde{N}\}.
\end{aligned}
$$

We want to show that this problem is equivalent to the problem

$$
\begin{aligned}
\text{(F-TRIP-R)} \qquad \min_{d,\zeta} \quad & c^T d + \sum_i \sum_{j>i} \zeta_{i,j} \\
\text{s.t.} \quad & -\zeta_{i,j} \leq u_i + d_i - u_j - d_j \leq \zeta_{i,j} \quad \forall i,j \in \{1,\dots,\tilde{N}\}, j > i, \\
& \sum_i |d_i| \leq \Delta, \\
& (\sum_i (u_i + d_i))(\sum_i (1 - u_i - d_i)) \leq \sum_i \sum_{j>i} \zeta_{i,j}, \\
& d_i + u_i \in [0,1] \quad \forall i \in \{1,\dots,\tilde{N}\},
\end{aligned}
$$

which we obtain by relaxing the integrality constraint and adding one non-convex quadratic inequality. We assume that $\alpha = 1$ as this is not a resticting as we are able to scale the cost function by $\frac{1}{\alpha}$ and obtain the same optimal points (with a scaled objective value). We define equivalence in the sense that a solution to one problem is also a solution for the other problem.

Theorem 8.13. *The problems* (F-TRIP) *and* (F-TRIP-R) *are equivalent in the sense that an optimal solution to one problem is also an optimal solution to the other problem with the same objective value.*

Proof. The constraint $(\sum_i (u_i + d_i))(\sum_i (1 - u_i - d_i)) \leq \sum_i \sum_{j>i} \zeta_{i,j}$ is fulfilled by any feasible point for (F-TRIP) as in the binary case this term is just the product of the cardinality of the set of nodes set to 0 and the cardinality of the set of nodes set to 1. Thus, we only need to show that (F-TRIP-R) only has integer-valued optimal solutions to prove the equivalence of the problems.

A major insight needed for the proof is that any optimal point for either problem fulfills the inequality $(\sum_i (u_i + d_i))(\sum_i (1 - u_i - d_i)) \leq \sum_{i>j} \sum_j \zeta_{i,j}$ with equality. The inequality

$$|u_i + d_i - u_j - d_j| \leq (u_i + d_i)(1 - (u_j + d_j)) + (u_j + d_j)(1 - (u_i + d_i)) \tag{8.2}$$

follows from

$$u_i + d_i - u_j - d_j = (u_i + d_i)(1 - \frac{u_j + d_j}{u_i + d_i}) \leq (u_i + d_i)(1 - (u_j + d_j))$$

for $u_i + d_i \in (0, 1)$, $u_j + d_j \in [0, 1]$ and the analogous case with swapped ropes of i and j as well as by inspection of the case with binary-valued points. Summation over all those terms gives us that

$$\sum_i \sum_{j>i} |u_i + d_i - u_j - d_j| \leq (\sum_i (u_i + d_i))(\sum_i (1 - u_i - d_i)) \leq \sum_i \sum_{j>i} \zeta_{i,j}.$$

Because (F-TRIP-R) is a minimization problem and (8.2) holds, we obtain that for any feasible point $(\bar{d}, \bar{\zeta})$ with $(\sum_i (u_i + \bar{d}_i))(\sum_i (1 - u_i - \bar{d}_i)) < \sum_i \sum_{j>i} \bar{\zeta}_{i,j}$ we can find a feasible point $(\bar{d}, \tilde{\zeta})$ with $(\sum_i (u_i + \bar{d}_i))(\sum_i (1 - u_i - \bar{d}_i)) = \sum_i \sum_{j>i} \tilde{\zeta}_{i,j}$ with a better objective value. Thus, in the following proof we assume wlog that this inequality always holds with equality and introduce

$$\omega(u + d) \coloneqq (\sum_i (u_i + d_i))(\sum_i (1 - u_i - d_i))$$

for sake of clarity.

Let d^* be optimal for (F-TRIP-R) with the corresponding ζ^* values. We distinguish between three cases.

Case 1: We assume that there only exists one entry in d^* which is not integer-valued. Then $\sum_i |d_i^*| < \Delta \in \mathbb{N}$ directly follows. The function $\omega(u + d)$ behaves strictly concave with regards to changes in one entry, as the only quadratic term which changes if an entry d_k changes is $(u_k + d_k)(1 - u_k - d_k)$. The remaining terms only depend linearly on d_k or not at all. Thus, by rounding the entry to the integer value with the smaller objective value between the two choices, we improve the objective value while maintaining the feasibility. This is a contradiction to the optimality assumption.

Case 2: We assume that exactly one non-integer entry $d_k^* \geq 0$ with $u_k = 0$ and one non-integer entry $d_\ell^* \leq 0$ with $u_\ell = 1$ exist. If $\sum_i |d_i^*| < \Delta$, we can find an $\varepsilon > 0$ sufficiently small such that $\tilde{d}$ and $\hat{d}$, which have the same entries as d^* except for $\tilde{d}_k = d_k^* + \varepsilon$, $\tilde{d}_\ell = d_\ell^* - \varepsilon$, $\hat{d}_k = d_k^* - \varepsilon$ and $\hat{d}_\ell = d_\ell^* + \varepsilon$, are feasible. $\tilde{d}$ has

a higher capacity consumption compared to d^* while $\hat{d}$ uses less capacity. From the construction it follows that $u_k+d_k^*+u_\ell+d_\ell^* = u_k+\tilde{d}_k-\varepsilon+u_\ell+\tilde{d}_\ell+\varepsilon = u_k+\tilde{d}_k+u_\ell+\tilde{d}_\ell$ and the same argument holds for $\hat{d}$. Thus, we obtain that $\omega(u+d^*) = \omega(u+\tilde{d}) = \omega(u+\hat{d})$. The objective values can only differ in the linear term $c^T d$ which implies that **either** $\tilde{d}$ has an objective value less than or equal to the objective value of d^* **or** $\hat{d}$ has an objective value less than the objective value of d^*. Since the second case is a contradiction to the optimality assumption, we can assume without loss of generality that $\sum_i |d_i^*| = \Delta$ which also implies that $u_k + d_k^* = u_\ell + d_\ell^*$ due to $|d_k^*| + |d_\ell^*| = 1$.

We introduce $\bar{d}$ and $\underline{d}$ which have the same entries as d^* except that $\bar{d}_k = d_k^* + \varepsilon$, $\bar{d}_\ell = d_\ell^* + \varepsilon$, $\underline{d}_k = d_k^* - \varepsilon$ and $\underline{d}_\ell = d_\ell^* - \varepsilon$ with $\varepsilon = \min\{|d_k^*|, |d_\ell^*|\}$. The feasibility of both follows from the feasibility of d^*. We introduce the function $f(a) = a(\kappa - a)$ with a constant $\kappa > 0$. This function is strictly concave. For a given $\bar{a} > 2\varepsilon > 0$ it holds that

$$hr \coloneqq |f(\bar{a}) - f(\bar{a}+2\varepsilon)| \geq |f(\bar{a}) - f(\bar{a}-2\varepsilon)| =: hl$$

or

$$hl \geq hr.$$

The strict concavity implies

$$\begin{aligned} & f(\bar{a}-2\varepsilon) + f(\bar{a}+2\varepsilon) < 2f(\bar{a}) \\ \iff & f(\bar{a}-2\varepsilon) - f(\bar{a}) < f(\bar{a}) - f(\bar{a}+2\varepsilon) \end{aligned}$$

which in turn means that $f(\bar{a}+2\varepsilon) < f(\bar{a})$ holds if $hr \geq hl$, and $f(\bar{a}-2\varepsilon) < f(\bar{a})$ holds if $hl \geq hr$. This is visualized in Figure 8.2. If we set $\bar{a} = (\sum_i (u_i+d_i^*))(\sum_i (1-u_i-d_i^*))$ and $\kappa = \sum_i 1$, then we obtain that $f(\bar{a}) = \omega(u+d^*)$, $f(a+2\varepsilon) = \omega(u+\bar{d})$ and $f(\bar{a}-2\varepsilon) = \omega(u+\underline{d})$. Since the first term of the objective function behaves linearly, this implies that $\bar{d}$ or $\underline{d}$ has a lower objective value compared to d^* which is a contradiction to the optimality assumption.

Case 3: We now consider the case that d^* contains an arbitrary number of non-integer entries. We introduce the sets $Z = \{i : u_i = 0,\ d_i^* \notin \mathbb{Z}\}$ and $O = \{i : u_i = 1,\ d_i^* \notin \mathbb{Z}\}$. Let $\Delta_Z = \sum_{i \in Z} |d_i^*|$ and $\Delta_O = \sum_{i \in O} |d_i^*|$. We introduce $\tilde{d}$ with $\tilde{d}_i = d_i^*$ for all $i \in \{1, \dots, \tilde{N}\} \setminus (Z \cup O)$. We set the $\lfloor \Delta_Z \rfloor$ entries of $\tilde{d}$ with the lowest (most negative) entries of the cost vector c with regards to the index set Z to 1 and the entry of $\tilde{d}$ with the $\lceil \Delta_Z \rceil$ lowest cost vector entry to $\Delta_Z - \lfloor \Delta_Z \rfloor$. (Essentially the same procedure as the greedy procedure for the knapsack relaxation, see Section 5.3) Furthermore, we set the $\lfloor \Delta_O \rfloor$ entries of $\tilde{d}$ with the highest entries of the cost vector c with regards to the index set O to -1 and the entry of $\tilde{d}$ with the $\lceil \Delta_O \rceil$ highest cost vector entry to $-(\Delta_O - \lfloor \Delta_O \rfloor)$. The remaining entries are set to

0. We note that $\omega(u+\tilde{d}) = \omega(u+d^*)$ by the construction of $\tilde{d}$. Furthermore, it holds that $c^T\tilde{d} \leq c^T d^*$. Thus, we have found a feasible point with at most one positive and one negative non-integer point which has an objective value lower or equal to d^*. This point is not optimal by the proof of the second case (or first case if either O or Z is empty).

Combining all cases we obtain that the optimal solution to (F-TRIP-R) has to be integer-valued. The objective values coincide as the additional constraint does not influence integer-valued feasible points and thus the objective functions and remaining constraints are the same. □

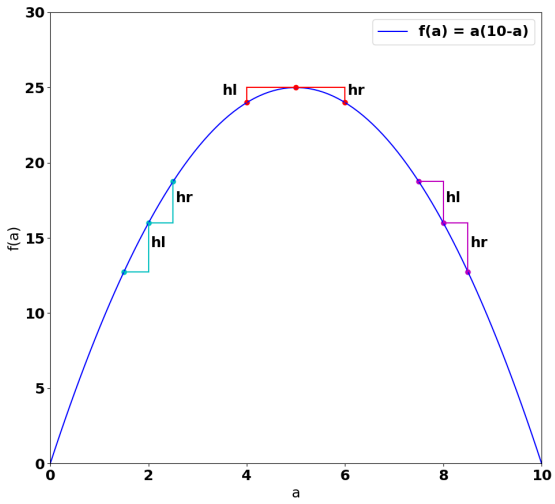

Figure 8.2: Visualization of the case distinctions. We see that the difference in objective values between the point with the lowest value and the middle point is larger than the difference between the remaining point and the middle point.

This reformulation of the problem does, however, not prove that it is solvable in polynomial time. The following proof shows that the problem (F-TRIP) is indeed polynomially solvable.

Theorem 8.14. *The problem* (F-TRIP) *can be solved in* $\mathcal{O}(\tilde{N}^2)$.

Proof. We have already seen that $(\sum_i(u_i+d_i))(\sum_i(1-u_i-d_i)) = \sum_i\sum_{j>i}\zeta_{i,j}$ for any optimal integer-valued point. The value of the sum of the jumps $\zeta_{i,j}$ is determined by the difference of the amount of capacity spent on nodes with $u_i = 0$ and the amount of capacity spent on nodes with $u_i = 1$. We denote by $Z \coloneqq \{i : u_i = 0\}$ and $O \coloneqq \{i : u_i = 1\}$. Furthermore, let $\Delta_0 \leq |Z|$ be the amount of capacity spent on nodes with $u_i = 0$ and by $\Delta_1 \leq |O|$ the amount of capacity spent on nodes with $u_i = 1$. There are at most 2Δ possible differences $\Delta_0 - \Delta_1$ as we note that depending

on the previous control only a smaller amount of capacity may be spent if there are not enough nodes with $u_i = 0$ or $u_i = 1$.

Let $k := \Delta_0 - \Delta_1 \in \mathbb{Z}$ be fixed. W.l.o.g. we assume that $k \geq 0$, otherwise, we switch roles. We construct an optimal solution as follows. We choose the cost vector entries with regards to the set Z with the k lowest values and set the corresponding entries in our constructed solution d^* to 1. We now choose the next lowest cost vector entry with regards to Z called c_Z and the next highest cost vector entry with regards to O called c_O. If $c_Z - c_O < 0$, we set $d^*_Z = 1$ and $d^*_O = -1$. We repeat this process until the used capacity would exceed Δ or there are no more unused entries in either of the sets. By construction it is obvious that for the given fixed difference and thus fixed sum of jumps this solution is optimal. If we compare the objective values of all such constructed solutions for the feasible differences, we obtain an optimal solution to (F-TRIP). In total, this can be done in $\mathcal{O}(2\Delta\tilde{N}) = \mathcal{O}(\tilde{N}^2)$. □

In the next theorem we will show that we can derive a most-violated cutting plane for the linear programming relaxation of (F-TRIP) with a previous control $u \equiv 0$ for which the whole domain is the fractional component. The coefficients are all either the negative minimum cut ratio or 1. We have already seen that the amount of jumps for the fully connected graph is given by the product of the number of nodes set to 0 and the number of nodes set to 1. Due to the concavity of the product, the minimum cut ratio is given by $\frac{(\tilde{N}-\Delta)\Delta}{\Delta} = (\tilde{N} - \Delta)$.

Theorem 8.15. *Let $u \equiv 0$. Let $d^* \equiv \frac{\Delta}{\tilde{N}}$. Let the corresponding ζ^* be given such that $\zeta^*_{i,j} = |d^*_i - d^*_j|$ for $i, j \in \{1, \ldots, \tilde{N}\}, i > j$ hold. Then the most-violated cutting plane of the form*

$$\kappa \leq \mu^T d + \sum_i \sum_{j>i} \zeta_{i,j} \tag{8.3}$$

for cutting off d^* with respect to the problem (F-TRIP) *is given by setting $\mu_i = -\rho = -(\tilde{N} - \Delta)$ for all $i \in \{1, \ldots, \tilde{N}\}$ and κ is the lowest value for the right hand side achieved by a feasible integer point for* (F-TRIP)*. In this context most-violated means that no other choice for κ and μ produces a larger gap $\kappa - (\mu^T d^* + \sum_i \sum_{j>i} \zeta^*_{i,j})$. In case of a tie, we select the choice which minimizes the ℓ^1 norm of μ.*

Proof. We note that the construction for κ always chooses the highest possible value, as higher values would in turn mean that feasible integer points are cut off. Lower values for κ would reduce the gap between integer points and d^*. We prove the result in two steps.

Case 1: We first turn to the case that all μ_i have the same value. We define two integer-valued feasible points. The first point is $d^0 \equiv 0$ while the second point $\bar{d}$ has exactly Δ many entries set to 1 with the remaining entries set to 0.

Subcase 1a: If $\mu \equiv -\rho$, then the left hand side of (8.3) is $\kappa = 0$ and the right hand side value for d^* is $-\Delta(\tilde{N} - \Delta) = -\rho\Delta$. By the construction of ρ it holds that $\bar{d}$ and d^0 are optimal points with regards to the right hand side of (8.3) with the value of 0. A feasible point which has more none-zero entries than d^0 but fewer than Δ many 1 entries has a value strictly larger than 0. Thus, the gap between d^* and any feasible integer point is at least $\rho\Delta$.

Subcase 1b: If $\mu \equiv \bar{\mu} < -\rho$ with $\bar{\mu} \in \mathbb{R}$, then the value for d^* is $\bar{\mu}\Delta$. Then $\bar{d}$ is optimal with regards to minimizing the right hand side with a value of $\bar{\mu}\Delta + \rho\Delta < 0$ which is the value for κ. Every feasible point with fewer entries set to 1 has a strictly larger value. So in this case, the gap between κ and the right hand side value for d^* is also given by $\rho\Delta$ but the absolute value of the left hand side is larger than in the first subcase and $|\bar{\mu}| > |\rho|$.

Subcase 1c: If $\mu \equiv \bar{\mu} > -\rho$ with $\bar{\mu} \in \mathbb{R}$, then the right hand side for d^* is given by $\bar{\mu}\Delta$. The only optimal point with regards to the right hand side is d^0 with a value of 0 which is the value for κ. Every other feasible point has a larger right hand side of the inequality. Thus, the gap is given by $-\bar{\mu}\Delta < \rho\Delta$.

In summary, in the case that all entries of μ are the same, $\mu \equiv -\rho$ is the optimal choice.

Case 2: Now let the μ_i take arbitrary values. We define $a \coloneqq \sum_i \frac{\mu_i}{\tilde{N}}$ as the average of the entries of μ. The right hand side of (8.3) for d^* is $a\Delta$. We determine the Δ many entries μ_i with the lowest values. We assume that the corresponding entries of $\bar{d}$ are set to 1. We define the average of this subset of the cost entries as $a^* = \sum_{i \in \{i | \bar{d}_i = 1\}} \frac{\mu_i}{\Delta}$. It follows directly that $a^* \leq a$ where equality holds iff all entries of μ are identical. The right hand side of (8.3) for $\bar{d}$ is $a^*\Delta + \rho\Delta$ which is an upper bound for κ. Thus, the difference is less than $\rho\Delta$ if not all of the values μ_i are identical. If they are identical, the first case applies.

In total, this shows that setting all entries of μ to $-\rho$ is optimal.

□

Remark 8.16. We note that we assumed that the whole domain is the fractional component. If that is not the case, it holds that the corresponding node-induced subgraph to the fractional component is still a fully-connected graph for which we can use the previous result.

Remark 8.17. The coefficients for the jump terms $\zeta_{i,j}$ are all 1 in (8.3). Changing some coefficients to values smaller than 1 would decrease the gap between the feasible integer-valued points and d^* (as $\bar{d}$ would have a lower right hand side compared to

before if constructed accordingly). Setting some coefficients to values larger than 1 would result in d^0 being the feasible integer-valued point with the lowest value for the right hand side. The gap would not change in consequence but this would increase the ℓ^1 norm of the coefficients for the right hand side.

This cutting plane can now be transferred to our setting for (TR-IP). If $\mathcal{G}$ was a fully connected graph instead of a subgraph of the grid, we could use that $|\partial U| = |V \setminus U||U|$ for any subset $U \subset V$ and thus that the minimum cut ratio of $\mathcal{G}$ would be $\tilde{\rho} = |V| - \Delta_r = |G| - \Delta_r$ as seen above. To transform $\mathcal{G}$ into a fully connected graph we would have to add quadratically many edges and thus quadratically many variables to our original problem formulation of (TR-IP) in order to model all new edges. This is not viable as it would significantly increase the computation time for the linear programming relaxation. Instead, we return to our original grid structure by replacing the new edges by weights on the original edges. Assume that the edge $e = (v_k, v_l)$ is added to construct the fully connected graph. If e is a cut edge, then every path from v_k to v_l is also cut. We increase the weight by 1 for all edges along a path from v_k to v_l in the original subgraph. We choose the path randomly among the set of shortest paths between the nodes. If we repeat this for every added edge, then the sum of weighted jumps for the original edges bounds the amount of jumps for the edges of the fully connected graph from above. We denote the resulting weights by $w^{\beta}_{i,j}$ and $w^{\gamma}_{i,j}$ respectively. This procedure is visualized in Figure 8.3.

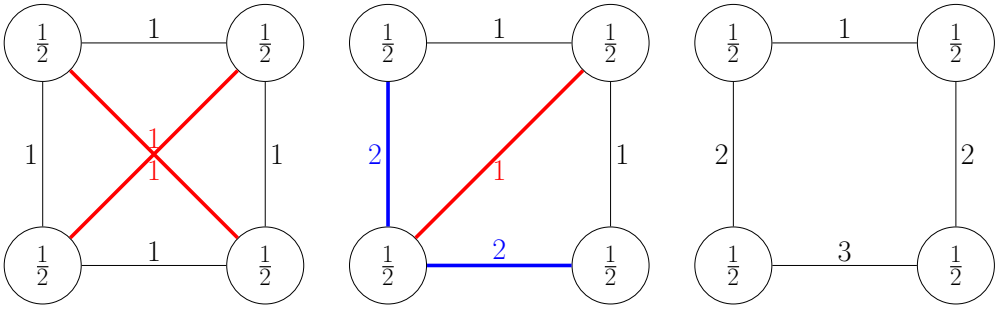

Figure 8.3: We replace edges of the fully connected graph by weights on the grid edges.
(Left) Fully connected graph where each edge has a weight of 1
(Middle) The edge is replaced by the blue path and the weights along the path are increased by 1.
(Right) All added edges are deleted and the weights are adjusted accordingly.

The success of the cutting plane also depends on M. We need to choose M just large enough that the inequality holds for feasible Ξ-valued points with a higher capacity use on the fractional component. This directly leads to the following theorem.

Theorem 8.18 (Theorem 8 in Section 5.1 in [74]). *Let $\Xi = \{0, 1\}$. Let Δ_{out} be a fixed integer between 0 and Δ with $\Delta_r = \Delta - \Delta_{out}$. Let G, H be disjoint sets of index*

pairs, $|H| = \Delta_{out}$ *and* G *be a connected component with the same value* $u_{i,j}$ *for every* $(i,j) \in G$*. Let* $\mathcal{G}(V,E)$ *be the graph corresponding to* G *and* $\tilde{\rho} = |G| - \Delta_r > 0$*. Every feasible point* $(d, \delta, \beta, \gamma)$ *of* (IP) *fulfills the inequality*

$$\tilde{\rho} \sum_{\substack{(i,j) \\ \in G}} \delta_{i,j} \leq \sum_{\substack{((i,j),(i+1,j)) \\ \in G \times G}} w^{\beta}_{i,j} \beta_{i,j} + \sum_{\substack{((i,j),(i,j+1)) \\ \in G \times G}} w^{\gamma}_{i,j} \gamma_{i,j} + M(\Delta_{out} - \sum_{\substack{(i,j) \\ \in H}} \delta_{i,j}) \tag{8.4}$$

for all $M \geq M_0$*, where*

$$\begin{aligned} M_0 :=& \tilde{\rho} - \min_{U \subset G, \Delta \geq |U| > \Delta_r} \frac{(|G| - |U|)|U| - (|G| - \Delta_r)\Delta_r}{|U| - \Delta_r} \\ =& \tilde{\rho} - \frac{(|G| - \min\{|G|, \Delta\}) \min\{|G|, \Delta\} - (|G| - \Delta_r)\Delta_r}{\min\{|G|, \Delta\} - \Delta_r}. \end{aligned}$$

Proof. Inequality (8.4) follows from the considerations above and using a big-M formulation to ensure that the inequality also holds for feasible integer points which spent less than Δ_{out} capacity on H. If $\Delta_{out} = 0$, it follows from the definition of the minimum cut ratio that

$$\tilde{\rho} \sum_{(i,j) \in G} \delta_{i,j} \leq \sum_{\substack{((i,j),(i+1,j)) \\ \in G \times G}} w^{\beta}_{i,j} \beta_{i,j} + \sum_{\substack{((i,j),(i,j+1)) \\ \in G \times G}} w^{\gamma}_{i,j} \gamma_{i,j}.$$

If $\Delta_{out} > 0$, we need to introduce the big-M term to account for feasible integer points which spend more capacity on G by spending less capacity on H. For the valid choice of M we analyze both terms in its definition. The first term negates the additional effect of the first term on the left-hand side in the inequality (8.4) if more than Δ_r capacity is used on the fractional component. The coefficient $\tilde{\rho}$ was derived from the minimum cut ratio with a capacity bound of Δ_r. A higher bound would imply a better minimum cut ratio. The term would thus no longer linearly underestimate the amount of jumps.

The second term ensures that the inequality remains valid as the amount of jumps changes as more than Δ_r capacity is used on the fractional component. We can interpret the amount of jumps $|\partial U| = |V \setminus U||U|$ as a concave function in $|U|$ meaning in the amount of capacity used on the fractional component. Thus, the construction of M ensures that we affinely underestimate the amount of jumps for a given capacity. The term $(|G| - \Delta_r)\Delta_r$ is the amount of jumps if Δ_r capacity is spend on the fractional component, while $(|G| - |U|)|U|$ is the amount of jumps if $|U|$ capacity is spend on the fractional component. The difference in capacity is given by $|U| - \Delta_r$.

In total, we construct a piecewise linear function (two pieces to be precise) mapping the capacity spent on G to an underestimation of the amount of jumps. The underestimation is actually tight for no capacity spent, Δ_r capacity spent or Δ capacity spent if all capacity is spent on G and H. This is visualized in Figure 8.4. □

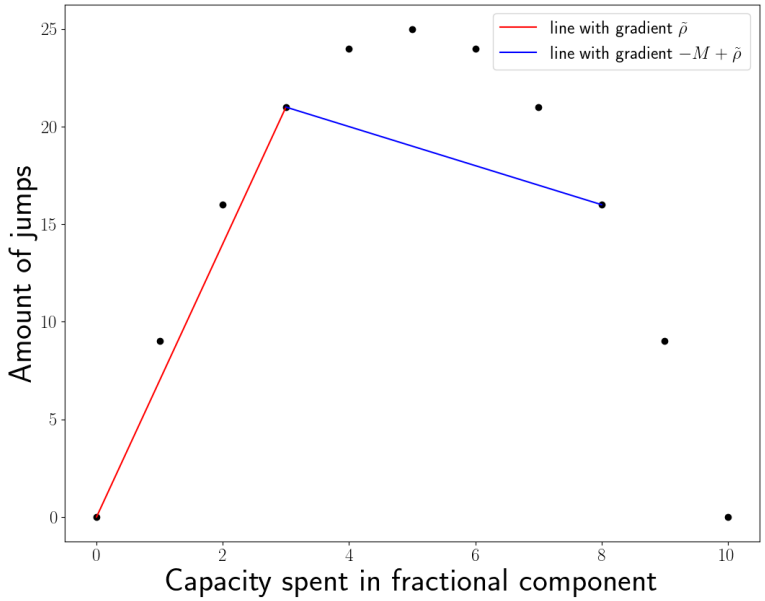

Figure 8.4: Visualization of the connection between the capacity used on the fractional component and the amount of jumps (cut edges) in the fully connected graph. For the example visualized above, the values are given by $\Delta = 8$, $\Delta_r = 3$ and $|G| = 10$. We can see that $\tilde{\rho}$ and M are chosen such that the amount of jumps is always affinely underestimated. Original figure in [74].

For the optimal solution to (LP) used for the construction, the right hand side of the equation is equal to 0 as all $\beta_{i,j}$ and $\gamma_{i,j}$ are equal to 0 and the sum of the $\delta_{i,j}$ regarding the subset H is exactly Δ_{out} by construction. The left side of the equation is strictly larger than 0 because $\tilde{\rho} > 0$ and $\sum_{(i,j)\in G} \delta_{i,j} > 0$. Thus, this solution is cut off by the constructed inequality improving the relaxation formulation.

Cutting plane derived from a bounding box

To obtain the previous cutting plane, we have used the fully connected graph extension of $\mathcal{G}$. It admits the drawback that the computational time to calculate the weights grows quadratically in the size of the fractional component. In the worst case, the fractional component and G are identical to the whole graph. Thus, for very large connected components the trade off between the compute time of the cutting plane and the run time reduction obtained from adding the cutting plane might not be worth it. Instead, we want to calculate a slightly different cutting plane with significantly lower compute time (linear vs. quadratic run time). For the construction we may need to include additional nodes outside of the fractional component.

The idea is that instead of only considering the jumps in the fractional component (or in a subset G), we find the smallest bounding box containing the component and add a cutting plane using the minimum cut ratio on this rectangular subgraph. We use the setting from the previous subsection and define $B \coloneqq \{(i,j) \in \Omega \mid \exists (i,k),(\ell,j) \in F\}$ as the smallest bounding box containing F for which we assume that the previous control in the nodes is 0 for now, see Remark 8.20 below. Furthermore, we define $H_B \coloneqq H \setminus B$ and $\Delta_B = \Delta - \sum_{(i,j)\in H_B} \delta^*_{i,j}$ and $\Delta_{out_B} = \sum_{(i,j)\in H_B} \delta^*_{i,j}$.

We recall that for the bounding box we can underestimate the amount of cut edges for a given used capacity K with the formula given in Lemma 6.2 which we already used for the NP-hardness conjectures. Note that in the original formulation of Lemma 6.2 our B was called G.

Theorem 8.19 (Theorem 9 in Section 5.1 in [74])**.** *Let $\Xi = \{0,1\}$. Let $\Delta_{out_B} \in \{0,\dots,\Delta\}$. Let B be set such that the corresponding graph is a $\tilde{n} \times \tilde{m}$ rectangular subgraph of the grid and with $u_{i,j} = 0$ for all $(i,j) \in B$. Let H_B be a set disjoint from B with $|H_B| = \Delta_{out_B}$. Then every feasible point of* (IP) *fulfills*

$$\kappa_\rho \sum_{(i,j)\in B} \delta_{i,j} \leq \sum_{\substack{((i,j),(i+1,j))\\ \in B\times B}} \beta_{i,j} + \sum_{\substack{((i,j),(i,j+1))\\ \in B\times B}} \gamma_{i,j} + M(\Delta_{out_B} - \sum_{(i,j)\in H_B} \delta_{i,j}).$$

with $\kappa_\rho \coloneqq \min_{0<K\leq\Delta-\Delta_{out_B}} \frac{\min\{\lfloor\sqrt{K}\rfloor+\lceil\sqrt{K}\rceil,\tilde{n},\tilde{m},\lfloor\sqrt{\tilde{n}\tilde{m}-K}\rfloor+\lceil\sqrt{\tilde{n}\tilde{m}-K}\rceil\}}{K}$ *and* $M \coloneqq \kappa_\rho + 2$.

Proof. Lemma 6.2 showed that κ_ρ is a lower bound for the amount of jumps in a $\tilde{n} \times \tilde{m}$ rectangular graph if at most $\Delta - \Delta_{out_B}$ nodes are set to 1. Thus, it follows that the inequality holds if M is chosen sufficiently large. In the proof of Lemma 6.2 it was shown that the minimal amount of jumps created if K nodes are set to 1 and the minimal amount of jumps created if $K+1$ nodes are set to 1 differ by at most 2. It follows that the first term in the definition of M negates the effect of the term on the left-hand side for values of K larger than $\Delta - \Delta_{out_B}$, while the second term accounts for the highest possible decrease in the amount of jumps which is no more than 2 for each additional node set to 1. □

The relevant computational demands are determining the bounding box and the capacity used by the solution to (LP) in the nodes of the bounding box. These demands are linear in the size of the bounding box. In the worst case the size of the bounding box is quadratic in the size of the fractional component, but is bounded by the size of the grid which ensures that the calculation of the cutting plane is significantly faster than the calculation of the previous cutting plane for large fractional components.

Remark 8.20. For the construction we have assumed that the previous controls in all nodes in the whole bounding box are 0. This is, however, not needed and only done for the sake of clarity of the argument. If c_2 is the number of nodes with a previous control of 1, the described cutting planes are valid if we instead use the formula $\min\{\lfloor\sqrt{K}\rfloor + \lceil\sqrt{K}\rceil, \tilde{n}, \tilde{m}, \lfloor\sqrt{\tilde{n}\tilde{m} - K - c_2}\rfloor + \lceil\sqrt{\tilde{n}\tilde{m} - K - c_2}\rceil\}$ to underestimate the amount of jumps and only consider $\delta_{i,j}$ on the left-hand side with $u_{i,j} = 0$. This is a direct consequence of applying the original formula in Lemma 6.2 with $\tilde{K} = K + n_c$ for $n_c \in [0, c_2] \cap \mathbb{N}$.

Remark 8.21. After applying a cutting plane, the resulting linear programming relaxation in the current node of the branching process might not have a solution with one fractional component anymore which would hinder the addition of further cutting planes of the above form. In practice, we often see that we obtain a solution with several fractional components which are all part of a connected subgraph. Thus, one could use the union of all fractional components as the component for the cutting plane construction. As we do have non-zero jump terms in the linear programming solution, it might not be cut off by the constructed cutting plane. A different approach is to choose one of the fractional components (preferably the largest) and take the capacity used on all the fractional components for this single component (and all capacity not used at all). If the size of this fractional component is larger than the amount of capacity, the above cutting planes are able to cut off the linear programming relaxation.

During the branching process of an integer programming solver employing a branch-and-bound approach, one often encounters nodes for which the previously added cutting planes are inactive. These nodes thus have only one fractional component and a cutting plane can be applied.

Extension to the non-binary case

For the cutting planes we have assumed a setting in which the set Ξ is binary. We now discuss how to extend the cutting planes to the case that Ξ is non-binary but the remaining setting stays the same, meaning Ξ is contiguous and G denotes the largest connected component with the same previous control value inside the fractional component. Without loss of generality we assume that the previous control values in G are 0. We first focus on the case that all capacity is spent inside the fractional component.

For the fully connected graph we have seen that the amount of jumps is equal to the product of entries set to 0 and entries set to 1 which is the same as the number of entries set to 0 times the capacity spend on G. The latter view also holds for the non-binary case. Let n be an arbitrary node in the fractional component for which the corresponding entry is not set to 0. Then the amount of capacity spend in n is equal to each jump height between n and any node with an entry of 0 in G. Summing

over all nodes, we obtain that $(|G| - k)k$ is a lower bound for any feasible integer point which spends k capacity on G. Thus, the formula still holds for the non-binary case and the coefficient is $|G| - \Delta$.

For the bounding box cutting plane the same argumentation holds. We discuss the case that $\Xi = \{0, 1, 2\}$ and the previous control $u|_B \equiv 0$ for the bounding box B. Let $\Delta < |B|$ (otherwise the coefficient derived from the minimum cut ratio is zero by design) and d be an arbitrary feasible integer-valued point. Let T be a maximal connected component with $d|_T \equiv 2$. Let O be the largest connected component which contains T with $d_i \in \{1, 2\}$ for every $i \in O$. Without loss of generality we assume that the other entries are 0 (which is fine because the components with different values are disjoint from O and T). We can view this integer assignment as the sum of two copies of the bounding box with binary-valued points $\tilde{d}$ and $\bar{d}$ defined as follows. For the first copy it holds that $\bar{d}_i \equiv 1$ if $d_i = 2$ for $i \in O$ and 0 otherwise, and for the second copy it holds that $\tilde{d}|_O \equiv 1$ and 0 otherwise. The sum of the two copies $\tilde{d} + \bar{d}$ produce the same sum of jump heights and the same capacity consumption, see Figure 8.5 for a visualization. The proof of Lemma 6.2 showed that for two smaller components we can find a larger component with the same amount of capacity use with a better (or equal) quotient of the sum of jump heights and capacity use. This is reflected in the formula of Lemma 6.2 which shows that a better quotient between the sum of jump heights and the capacity use can be achieved as the capacity use increases. Thus, we can find a feasible integer-valued point d^* with the same capacity consumption as d with $d_i^* \in \{0, 1\}$ for all $i \in B$ and a better (or equal) quotient of the sum of jump heights and capacity use (this needs $\Delta < |B|$ as assumed). Thus, we only need to consider the binary case here. Additionally, if Ξ contains integers larger than 2 the same argument applies. If Ξ contains integers smaller than 0 as well as all positive integers with the same absolute values and d is a feasible integer point, we can just choose d^* with $d_i^* = |d_i|$ to obtain a feasible point which is at least as good regarding the quotient as this does not increase any jump heights (but might reduce some). It follows that the set Ξ can be an arbitrary finite contiguous subset of the integers and the coefficient κ_ρ for the bounding box cutting plane remains valid. This also shows that the restriction that the previous control is equal to 0 on the whole bounding box can be relaxed to an arbitrary element of Ξ as long as it is the same for the whole bounding box. For a visualization see Figure 8.5.

We now discuss the case that not all capacity is spent inside of the fractional component. We have just shown that the coefficients except M can remain as for the binary case. Since M is derived from these values, M can be choosen as above for the binary case. However, the values $\Delta_{out} = |H|$ and $\Delta_{out_B} = |H_B|$ have to be changed. These values depended on the property that the solution to the linear relaxation used the maximal capacity consumption in the nodes since this is equivalent to spending capacity at all in the binary case. Thus, no feasible integer point could spend more capacity on H or H_B which ensured that the terms $M(\Delta_{out} - \sum_{(i,j)\in H} \delta_{i,j})$

and $M(\Delta_{out_B} - \sum_{(i,j)\in H_B} \delta_{i,j})$ are non-negative for any feasible integer point in the binary case. In the non-binary case the fractional component might spend less than the maximum and a feasible integer point might spend more capacity in the node. To ensure that this feasible point is not cut off, we consider the possible maximal capacity use in each node in which the solution to the linear programming relaxation spends capacity. Thus, we define $\Delta_{out} = \sum_{(i,j)\in H} \max_{\xi\in\Xi} |\xi - u_{i,j}|$ and $\Delta_{out_B} = \sum_{(i,j)\in H_B} \max_{\xi\in\Xi} |\xi - u_{i,j}|$ to prevent that any feasible integer point is cut off. We note that the solution of the linear programming relaxation might not be cut off anymore. Alternatively, one could calculate the cutting plane as if all capacity is spend on the fractional component. If $\Delta < |B|$ holds, the resulting cutting plane will cut off the solution to the linear programming relaxation. A combination is possible by only additionally considering the capacity spend on nodes in H or H_b for which the linear programming relaxation does not spend the maximal possible amount of capacity.

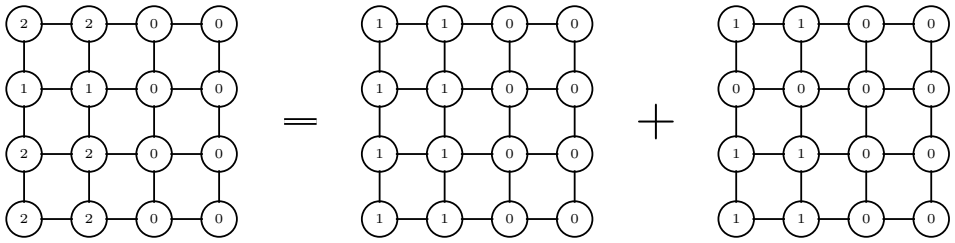

Figure 8.5: Visualization of the extension argument for the bounding box cutting plane to arbitrary contiguous Ξ.

Chapter 9

Improvements to the decomposition approach for the trust-region algorithm

The approach of the previous chapter focused on the individual (TR-IP) problem to reduce the computational demand. In this chapter we concern ourselves with the overall trust-region algorithm. A decomposition approach of the domain into patches with a corresponding adaptation of the trust-region algorithm was proposed in [6] and focuses on the trust-region algorithm overall. The resulting subproblems are of the form (TR-IP) but with some fixed variables and significantly smaller problem sizes. As a trade-off more (TR-IP) problems have to be solved in each step. The approach may result in a significant decrease in run time of the overall algorithm in two dimensions without significant changes in the quality of the solution, see the numerical results in [6]. We briefly restate the algorithm and relevant convergence results. For more details we refer the reader to [6]. Afterwards, we propose two different choices of patches to improve the decomposition approach further.

9.1 The decomposition approach

The underlying idea is to split the domain into a family of patches $\mathcal{D}$ and solve a trust-region subproblem on each individual patch. We state the assumptions on the patches and give the definition of stationarity on the patches.

Assumption 9.1 (Assumption 3.3 in [6])**.** Let $\mathcal{D} \subset \mathcal{P}(\Omega)$ be a finite, open cover of Ω.

Definition 9.2 (Definition 3.4 in [6])**.** Let $\mathcal{D} \subset \mathcal{P}(\Omega)$, $F : L^1(\Omega) \to \mathbb{R}$, and $\bar{u} \in \mathrm{BV}_\Xi(\Omega)$ satisfy the assumptions of Definition 3.4 and Assumption 9.1. Then $\bar{u}$ is

patch-stationary with respect to $\mathcal{D}$ if for all $D \in \mathcal{D}$ the identity (3.2) holds for all $\phi \in C_c^\infty(D, \mathbb{R}^d)$.

Theorem 9.3 (Theorem 3.5 in [6]). *Let $\mathcal{D} \subset \mathcal{P}(\Omega)$, $F : L^1(\Omega) \to \mathbb{R}$, and $\bar{u} \in \mathrm{BV}_\Xi(\Omega)$ satisfy the assumptions of Definition 9.2. Then $\bar{u} \in \mathrm{BV}_\Xi(\Omega)$ is stationary if and only if it is patch-stationary with respect to $\mathcal{D}$.*

The patches have to be choosen in such a way that the following assumption holds.

Assumption 9.4 (Assumption 5.1 in [6]). Let Ω be a bounded Lipschitz domain. Let the finite set of patches $\mathcal{D} \subset \mathcal{P}(\Omega)$ satisfy the following conditions.

1. For all $x \in \Omega$, there exists $r > 0$ and $D \in \mathcal{D}$ such that $\overline{B_r(x)} \subset\subset D$. (Patch overlap)

2. For all $D \in \mathcal{D}$, we assume that $D \in \mathcal{D}$ is a bounded Lipschitz domain. (Patch regularity)

The resulting trust-region subproblems for a given patch D are given by

$$\text{(P-TR)} \qquad \begin{aligned} \min_{d \in L^2(\Omega)} \ & (\nabla F(u), d)_{L^2(\Omega)} + \alpha \mathrm{TV}(u+d) - \alpha \mathrm{TV}(u) \\ \text{s.t.} \ & u(x) + d(x) \in \Xi \text{ for a.a. } x \in D, \\ & \|d\|_{L^1(\Omega)} \leq \Delta, \\ & d(x) = 0 \text{ for a.a } x \in \Omega \setminus D. \end{aligned}$$

We note that all the control entries outside of the individual patch are fixed to the value of the previous control. For the resulting integer programs after discretization we only need to consider the fixed entries which are adjacent to non-fixed entries with respect to the underlying grid as the remaining fixed entries do not effect the optimization.

The patch-based trust-region algorithm from [6] is depicted in Algorithm 4 and consists of an outer loop with two inner parts each containing further loops. The first part determines acceptable step candidates for the patches from the solutions to the trust-region subproblems. The second part employs a greedy approach to combine the step candidates with the goal to obtain a combination superior to any individual step candidate.

In detail, the algorithm starts with the initialization of the set of acceptable step candidates $\mathcal{A} = \emptyset$ and of the working set $\mathcal{W} = \{(0, D) | D \in \mathcal{D}\}$. Afterwards, the outer loop starts. The first part contains a for-loop (with iteration number k) which in itself contains another loop iterating over all patches D which have not produced an acceptable step candidate but can still produce a solution to the trust-region subproblem with a greater actual reduction than any of the previously calculated

acceptable step candidates. For each patch a trust-region subproblem with trust-region radius $\Delta^0 2^{-k}$ is solved and returns a trial candidate $\tilde{u}^{n,k,D}$. The predicted and actual reductions for the trial candidate are calculated which results in one of three cases.

Case 1: If the predicted reduction is positive and the actual reduction is larger than a given fraction of the predicted reduction, the trial candidate is added to the set of acceptable step candidates. The patch will not be considered in the next iteration of the inner loop.

Case 2: The second case applies if the first case does not hold but the predicted reduction is positive and the best actual reduction achieved by any of the acceptable step candidates is smaller than the upper bound for the actual reduction of the patch given by the sum of the predicted reduction and the product of $L_{\nabla F}$ (see Assumption 3.1) and the trust-region radius. This implies that a later iteration may still produce an acceptable step candidate with a better actual reduction than the actual reduction of any acceptable step candidate found so far. Thus, the patch is considered in the next iteration.

Case 3: If the other two cases do not hold, the patch is not considered in the next iteration.

The second part first checks if the set of acceptable step candidates is empty. If so, the current iterate u^n is stationary and the algorithm terminates. Otherwise, a greedy approach is used to determine the next iterate. Based on the actual reduction of the step size candidates the current iterate is updated on the corresponding patch until the objective value is no longer decreased. At least one patch is updated as the actual reduction of each acceptable step candidate is positive. Afterwards, the outer loop starts anew.

For the convergence of the algorithm the following theorem was proven in [6].

Theorem 9.5 (Theorem 5.8 in [6])**.** *Let Assumptions 3.1, 9.1 and 9.4 hold. Let $\{u^n\}_{n\in\mathbb{N}}$ be the sequence of iterates produced by Algorithm 4. Then one of the following mutually exclusive outcomes holds:*

1. *The sequence $\{u^n\}_{n\in\mathbb{N}}$ is finite and the final element u^N for some $N \in \mathbb{N}$ satisfies the following. For all $D \in \mathcal{D}$ there exists $k \in \mathbb{N}$ such that u^N solves P-TR with input $u^N, \nabla F(u^N), D$ and $\Delta^0 2^{-k}$. In particular, u^N is stationary (as $\nabla F(u^N) \in C(\bar{\Omega})$ follows from the assumptions).*

2. *The sequence $\{u^n\}_{n\in\mathbb{N}}$ is finite and the final element u^N for some $N \in \mathbb{N}$ satisfies the following. The loop over k that begins in Line 3 does not terminate*

Algorithm 4 Sequential linear integer programming method with greedy patch updates from [6]

Input: F satisfying the assumptions stated in [6] with smoothness constant $L_{\nabla F}$, $\Delta^0 > 0$, $u^0 \in \mathrm{BV}_\Xi(\Omega)$, $\theta \in (0,1)$, set of patches $\mathcal{D}$.

for $n = 0, 1, 2, \ldots$ **do**
  Set $\mathcal{A} \leftarrow \emptyset$, $\mathcal{W} \leftarrow \{(0, D) \,|\, D \in \mathcal{D}\}$.
  for $k = 0, 1, \ldots$ **do**
    while $(k, D) \in \mathcal{W}$ **do**
      $\tilde{u}^{n,k,D} \leftarrow$ minimizer of TR subproblem on D with $(u^n, \nabla F(u^n), D, \Delta^0 2^{-k})$.
      $\mathrm{pred}^{n,k,D} \leftarrow (\nabla F(u^n), u^n - \tilde{u}^{n,k,D})_{L^2} + \alpha \mathrm{TV}(u^n) - \alpha \mathrm{TV}(\tilde{u}^{k,n,D})$
      $\mathrm{ared}^{n,k,D} \leftarrow F(u^n) + \alpha \mathrm{TV}(u^n) - F(\tilde{u}^{n,k,D}) - \alpha \mathrm{TV}(\tilde{u}^{n,k,D})$
      if $\mathrm{ared}^{n,k,D} \geq \theta \, \mathrm{pred}^{n,k,D}$ **and** $\mathrm{pred}^{n,k,D} > 0$ **then**
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{(k, D)\}$.
      else if $\mathrm{pred}^{n,k,D} > 0$
      and $\max_{(\tilde{k},\tilde{D}) \in \mathcal{A}} \mathrm{ared}^{n,\tilde{k},\tilde{D}} < \mathrm{pred}^{n,k,D} + L_{\nabla F} \Delta^0 2^{-k}$ **then**
        $\mathcal{W} \leftarrow \mathcal{W} \cup \{(k+1, D)\}$
      end if
      $\mathcal{W} \leftarrow \mathcal{W} \setminus \{(k, D)\}$.
    end while
    if $\mathcal{W} = \emptyset$ **then**
      break
    end if
  end for
  if $\mathcal{A} = \emptyset$ **then**
    return (u^n is stationary).
  end if
  $\bar{u}^n \leftarrow u^n$.
  $\bar{j}^0 \leftarrow F(u^n) + \alpha \mathrm{TV}(u^n)$.
  while $\mathcal{A} \neq \emptyset$ **do**
    $\bar{k}, \bar{D} \leftarrow \arg\max\{\mathrm{ared}^{n,k,D} \,|\, (k, D) \in \mathcal{A}\}$
    $\tilde{u}^n \leftarrow \bar{u}^n \chi_{\Omega \setminus \bar{D}} + \chi_{\bar{D}} \tilde{u}^{n,\bar{k},\bar{D}}$
    $\bar{j} \leftarrow F(\tilde{u}^n) + \alpha \mathrm{TV}(\tilde{u}^n)$
    if $\bar{j} < \bar{j}^0$ **then**
      $\bar{u}^n \leftarrow \tilde{u}^n$, $\bar{j}^0 \leftarrow \bar{j}$, $\mathcal{A} \leftarrow \mathcal{A} \setminus \{(\bar{k}, \bar{D})\}$.
    else
      break
    end if
  end while
  $u^{n+1} \leftarrow \bar{u}^n$
end for

finitely. In particular, u^N is stationary (as $\nabla F(u^N) \in C(\bar{\Omega})$ follows from the assumptions).

3. *The sequence* $\{u^n\}_{n\in\mathbb{N}}$ *has a weak-* accumulation point in* $\mathrm{BV}(\Omega)$*. All weak-* accumulation points are in* $\mathrm{BV}_\Xi(\Omega)$*. A weak-* accumulation point* $\bar{u}$ *(which satisfies* $\nabla F(u^N) \in C(\bar{\Omega})$ *by assumptions) is stationary.*

We have already seen this result in Theorem 3.6 for the original trust-region algorithm which was proven by [73]. For the case of a single patch the Algorithm 4 turns into the original trust-region algorithm (Algorithm 1).

9.2 The newly proposed decompositions

We propose two shapes for the patches to improve the decomposition approach. The first approach relies on the insights presented in Chapter 7 regarding the fractional component. The computational results in [74] suggest that there is a correlation between the size of the fractional component in the root linear program and the time needed to solve the corresponding integer program. Thus, we want to choose a decomposition which minimizes the sizes of the fractional components which occur or which stops the occurence all together. The idea is to decompose the domain in such a way that the fractional components for the linear programming relaxations corresponding to the trust-region subproblems on the whole domain are divided as best as possible. In principle, the position of the fractional component could change with each new trust-region subproblem and choosing a decomposition based on any individual problem instance would not yield the desired effect. In practice, it seems to be common that the fractional component tends to be in the same general vicinity for most instances of an execution of the trust-region algorithm. Thus, we first run the trust-region algorithm without a decomposition and determine the first integer program which is not solved instantly. We calculate the fractional component of the linear programming relaxation and choose the center point of the bounding box around the fractional component as the point for our decomposition. We add the necessary overlap to each patch and furthermore ensure that every patch adheres to a minimum size (in each coordinate). We note that for the theoretical foundation from [6] to hold we cannot choose the dimensions of the patches too small.

The second decomposition approach we propose based on the insights gained so far is to switch from square shapes to stripes. In the original decomposition approach rectangular-shaped patches of the same size were used. If the original domain is square-shaped like in the numerical experiments in [6], this means that the domain is split into overlapping square-shaped patches. The insights from [96] as well as Section 8.1 regarding the one-dimensional case and [74] as well as Section 8.3 regarding the two-dimensional case support the notion that an underlying structure closer to the one-dimensional case is desirable. For an $N \times M$ grid a one-dimensional structure can be achieved by fixing a certain subset of the nodes with size $\min\{N-1, M-1\} \max\{\lfloor \frac{N}{2} \rfloor, \lfloor \frac{M}{2} \rfloor\}$, which is visualized in Figure 9.1. Thus,

we propose to decompose the domain into vertical or horizontal stripes to achieve rectangular structures where either N or M are significantly smaller than its counter part. Additionally, the connected segments with the same value tend to have shapes closer to squares. This can be explained by the fact that this shape has a better ratio of the volume to the boundary. Thus, we expect the decomposition into stripes to also benefit from a better division of the fractional components in a similar manner to the first proposed decomposition.

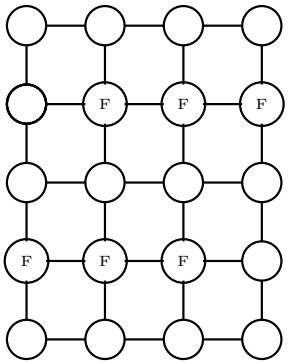

Figure 9.1: The nodes marked with an F are the fixed nodes in the grid. The remaining nodes can be solved with a shortest path approach for a graph with pseudo-polynomial size.

Chapter 10

Numerical experiments

In this section we will discuss the numerical results for several test problems for the one-dimensional and two-dimensional case. The chapter is split into two sections. In Section 10.1 we state the numerical results already published in [96] and [74] which focus on the individual (TR-IP). Afterwards in Section 10.2, we will provide new numerical results for the two-dimensional case but focus on the overall run time of the trust-region algorithm. These results will extend the results from [74] as now the decomposition approach is combined with the primal heuristic, the branching priority and the cutting planes.

10.1 Numerical results for the individual (TR-IP) problems

The results in [96] deal with the one-dimensional case to validate the proposed graph-based approaches. The two-dimensional case is discussed in [74] and in the numerical experiments we compare the run time of the integer programming solver without any of the proposed measures to the runtime of the integer programming solver with the primal heuristic, the branching priority and the cutting planes.

10.1.1 Results for the one-dimensional case

In order to assess the run times of the proposed graph-based computations, a parameterized instance of (IOCPs) is provided in [96]. The SLIP algorithm from [67] is run on discretizations of them, thereby generating instances of the integer programming problems corresponding to (1D-w), which are then solved with

(Astar) the A^* algorithm with a minimal variation (see [96] for details)

(TOP) the topological sorting algorithm described in [30], and

(SCIP) the general purpose IP solver SCIP, see [42].

The distribution of run times of the three different algorithmic approaches for solving the generated instances of (IP) is compared.

The parameterized IOCP is an integer optimal control problem that is governed by a steady heat equation in one spatial dimension. The SLIP algorithm was run on 25 discretized instances that differ in the choice of the value for the penalty parameter α (five different choices) and the discretization constant N (five different choices). We note that another parameterized IOCP is given in the Online Supplement to [96] which we did not include as it did produce similar results such that we decided against including it here and instead only refer the reader to it here.

The benchmark problem

We consider the class of IOCPs

$$\text{(SH)} \qquad \begin{aligned} \min_{u,v} \quad & \frac{1}{2}||v-\bar{v}||^2_{L^2(-1,1)} + \alpha \mathrm{TV}(u) \\ \text{s.t.} \quad & -\frac{d}{dt}(\varepsilon(t)\frac{dv}{dt}(t)) = f(t) + u(t) \text{ for a.a. } t \in (-1,1), \\ & v(t) = 0 \text{ for } t \in \{-1,1\}, \\ & u(t) \in \Xi = \{-2,\dots,23\} \subset \mathbb{Z} \text{ for a.a. } t \in (-1,1), \end{aligned}$$

which leans on [64] page 112 ff. with the choices $\varepsilon(t) = 0.1\chi_{(-1,0.05)}(t) + 10\chi_{[0.05,)}(t)$, $f(t) = e^{-(t+0.4)^2}$, and $\bar{v}(t) = 1$ for all $t \in [-1,1]$. The problem is rewritten in the equivalent reduced form

$$\begin{aligned} \min_{u} \quad & \frac{1}{2}\|Su + v_f - \bar{v}||^2_{L^2(-1,1)} + \alpha \mathrm{TV}(v) \\ \text{s.t.} \quad & u(t) \in \Xi = \{-2,\dots,23\} \subset \mathbb{Z} \text{ for a.a. } t \in (-1,1), \end{aligned}$$

where the function $S : L^2(-1,1) \to L^2(-1,1)$ denotes the linear solution map of the boundary value problem that constrains (SH) for the choice $f = 0$ and $v_f \in L^2(-1,1)$ denotes the solution of the boundary value problem that constrains (SH) for the choice $u = 0$. With this reformulation and the choice $F(x) \coloneqq \frac{1}{2}\|Su + v_f - \bar{v}||^2_{L^2(-1,1)}$ for $u \in L^2(-1,1)$, the required problem formulation (IOCP) is obtained, which gives rise to Algorithm 1 and the corresponding subproblems.

Implementation

We compute the discretization of the least squares term, the PDE, and the corresponding derivative of F for N discretization intervals with the open source package Firedrake, see [85]. We use a piecewise constant ansatz for the control and CG1 methods for the states. Note that the derivative is computed in Firedrake with the help of so-called adjoint calculus in a *first-discretize, then-optimize* manner, see [58]. Because we have a uniform discretization, we obtain $\sigma_i = 1$ for all $i \in [N]$. Thus,

we obtained a problem of the form (IP) and not of the more general form (1D-w). Regarding the solution of the generated subproblems, we have implemented both (Astar) and (TOP) in C++. For the solution approach (SCIP) we employ the SCIP Optimization Suite 7.0.3 with the underlying LP solver SoPlex, see [42], to solve the IP formulation (IP).

Computational setup

We run Algorithm 1 for all combinations of the parameter values $\alpha \in \{10^{-3}, \dots, 10^{-7}\}$ and uniform discretizations of the domain $(-1, 1)$ into $N \in \{512, \dots, 8192\}$ intervals. The number of intervals coincides with the problem size constant N in the resulting IPs of the form (IP). For all runs of Algorithm 1 we use the reset trust-region radius $\Delta^0 = \frac{1}{8}N$ and the step acceptance ratio $\theta = 0.1$.

On each of these 25 discretized instances of (SH), we run our implementation of Algorithm 1 with three different initial iterates u^0, thereby giving a total of 75 runs of Algorithm 1. Regarding the initial iterates u^0, we make the following choices:

1. we compute a solution of the continuous relaxation, where Ξ is replaced by $\operatorname{conv} \Xi = [-2, 23]$ and α is set to zero, with Scipy's implementation of limited memory BFGS with bound constraints, see [100] and [69], and round it to the nearest element in Ξ on every interval,

2. we set $u^0(t) = 0$ for all $t \in (-1, 1)$, and

3. we compute the arithmetic mean of the two previous choices and round it to the nearest element in Ξ on every interval.

For all IOCP instances with $N \in \{512, \dots, 2048\}$ we prescribe a time limit of 240 seconds for the IP solver but note that almost all instances require only between a fraction of a second and a single digit number of seconds compute time for global optimality so that the time limit is only reached for a few cases.

For higher numbers of N the run times of the subproblems are generally too long to be able to solve all instances with a meaningful time limit for the solution approach (SCIP). In order to assess the run times for (SCIP) in this case as well we draw 50 instances of (IP) from the pool of all generated subproblems both for $N = 4096$ and $N = 8192$ and solve them with a time limit of one hour each. Moreover, we do the same with the 50 instances of (IP) for which the solution approach (Astar) has the longest run times.

We note that a run of Algorithm 1 may return different results depending on which solution approach is used for the trust-region subproblems even if all trust-region subproblems are solved optimally because the minimizers need not to be unique and different algorithms may find different minimizers. In order to be able to compare the performance of the algorithms properly we record the trust-region

subproblems that are generated when using (Astar) as subproblem solver and pass the resulting collection of instances of (IP) to the solution approaches (TOP) and (SCIP).

A laptop computer with an Intel(R) Core i7(TM) CPU with eight cores clocked at 2.5 GHz and 64 GB RAM serves as the computing platform for all of our experiments.

Numerical results

We provide run times for each of the three algorithmic solution approaches (Astar), (TOP), and (SCIP). The 75 runs of Algorithm 1 on instances of (SH) generate 30440 instances of (IP) in total, see Table 10.1.

Table 10.1: Number of instances of (IP) generated by the runs of Algorithm 1 on the different discretizations and parameterizations of (SH). Original table in [96].

α \ N	512	1024	2048	4096	8192	$\sum$
10^{-7}	2055	2960	3692	4645	3878	17230
10^{-6}	1727	1291	639	507	1306	5470
10^{-5}	310	124	339	524	1018	2315
10^{-4}	365	296	427	345	986	2419
10^{-3}	303	597	583	675	848	3006
$\sum$	4760	5268	5680	6696	8036	30440

We first analyze the recorded run times of all solution approaches with respect to the value N. The run times of (Astar) and (TOP) turn out to be generally much lower than those of (SCIP) and we then analyze their run times in more detail with respect to the product of N and Δ and the number of nodes in the graph. Finally, we compare the run time and objective values achieved by Algorithm 1 using (Astar) with those obtained for the combinatorial integral approximation and a hybridization of both methods.

Run time with respect to N All instances of (IP) are solved in a couple of seconds with the solution approaches (Astar) and (TOP). The solution approach (SCIP) solves almost all instances for (SH) within the prescribed time limits. The exceptions are one instance with $N = 1024$ (out of 5268), 14 instances with $N = 2048$ (out of 5680), and two instances with $N = 4096$ (out of 100). For (Astar), the mean run time to solve the (IP) instances generated for (SH) increases from 0.016 s to 2.1 s over the increase of N from 512 to 8192. For (TOP), the mean run time starts from the lower value 0.015 s at $N = 512$, surpasses the mean run time of (Astar) for $N = 1024$ and increases to the higher value 4.4 s for $N = 8192$, all other things

being equal. For (SCIP), the mean run time increases from 0.51 s for $N = 512$ to 4.7 s for $N = 2048$. For $N = 4096$, the mean run time of (SCIP) is 210 s for the fifty randomly drawn instances and 128 s for the fifty instances, for which (Astar) has the highest run times. For $N = 8192$, the mean run time of (SCIP) is 56 s for the fifty randomly drawn instances and 598 s for the fifty instances, for which (Astar) has the highest run times. We observe qualitatively similar trends when considering median and shifted geometric mean instead of the mean, see Table 10.2.

Table 10.2: Mean (top), median (middle) and shifted geometric mean (shift of 10, bottom) run times in seconds of the solution process for the IPs of the form (IP) for different solution algorithms and problem sizes. The values for the IPs generated by running Algorithm 1 on the different discretizations and parameterizations of (SH) are tabulated. The smallest mean run time per value of N is written in bold type. For the algorithm (SCIP) the mean, median and geometric mean are computed over all instances (all) for $N \in \{512, 1024, 2048\}$ and over 50 randomly drawn instances (random) as well as the 50 instances with the highest run time of (Astar) (highest) for $N \in \{4096, 8192\}$. Original table in [96].

N	(Astar)	(TOP)	(SCIP)		
			all	random	highest
512	0.016	**0.015**	0.511	-	-
1024	**0.037**	0.071	0.901	-	-
2048	**0.100**	0.304	4.698	-	-
4096	**0.476**	1.220	-	209.751	128.430
8192	**2.122**	4.374	-	56.023	597.785

N	(Astar)	(TOP)	(SCIP)		
			all	random	highest
512	0.016	**0.005**	0.130	-	-
1024	0.033	**0.032**	0.292	-	-
2048	**0.076**	0.166	1.342	-	-
4096	**0.266**	0.820	-	4.014	88.143
8192	**0.861**	1.669	-	8.281	554.032

N	(Astar)	(TOP)	(SCIP)		
			all	random	highest
512	0.016	**0.015**	0.476	-	-
1024	**0.037**	0.070	0.766	-	-
2048	**0.100**	0.296	2.952	-	-
4096	**0.465**	1.144	-	19.524	62.144
8192	**1.876**	3.576	-	16.682	352.537

We note that (SCIP) did not solve two of the randomly drawn instances for $N = 4096$ within a one hour time limit, thereby affecting the mean significantly (it would be around 68 s without these two instances).

We illustrate the distribution of the run times for the different solution approaches and the different values of N with violin plots in Figure 10.1.

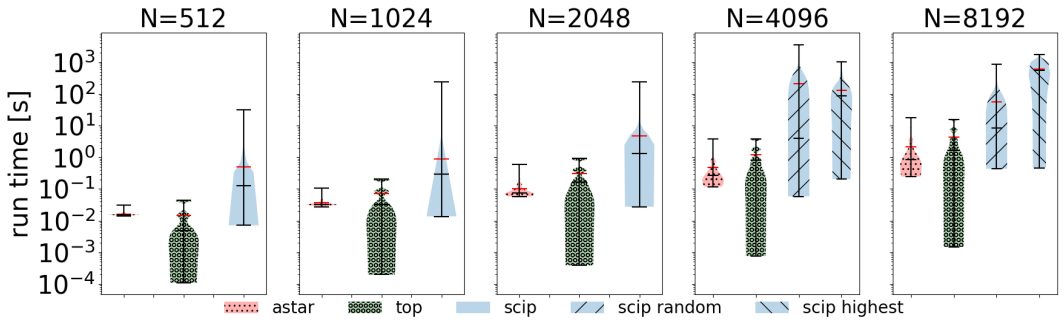

Figure 10.1: Distribution of the run times for the solution of the generated instances of (IP) with (Astar) (left), (TOP) (center), and (SCIP) (right) with range limits and mean (median) marked with red (black) strokes for the subproblems originating from (SH). For $N \in \{4096, 8192\}$ the results of the solution approach (SCIP) are split into the results for the randomly chosen instances ("scip random", center right) and the instances on which (Astar) has the highest run times ("scip highest", right). Original figure in [96].

Run times of the graph-based approaches based on graph size Because Ξ in (IP) depends only on the superordinate IOCP and does not vary over a run of Algorithm 1, we consider Ξ as fixed. Then the complexity of (TOP) depends linearly on the number of nodes and edges in the graph and thus the product of N and the trust-region radius Δ. Moreover, (TOP) cannot skip nodes or terminate early. In contrast to this, (Astar) can make use of the heuristic function and the preprocessing. Therefore, we assess the run times of (TOP) and (Astar) with respect to the problem size of (IP) measured as $N\Delta$.

We observe that the mean run times produced by (TOP) follow approximately a linear trend with respect to $N\Delta$ starting from very low values at the order of 10^{-4} s for $N\Delta \approx 10^4$ increasing to values at the order of 10^1 s for $N\Delta \approx 2 \cdot 10^8$. In contrast to this, the mean run times of (Astar) start at the order of 10^{-2} s for $N\Delta \approx 10^4$, follow a generally shallower but (at first impression less linear) trend, and are about an order of magnitude lower for the highest values of $N\Delta$. In particular, the run time of (TOP) starts exceeding the run time of (Astar) between $N\Delta = 10^6$ and $N\Delta = 10^7$ for the subproblems generated. The mean run times over the different values of $N\Delta$ are plotted in Figure 10.2.

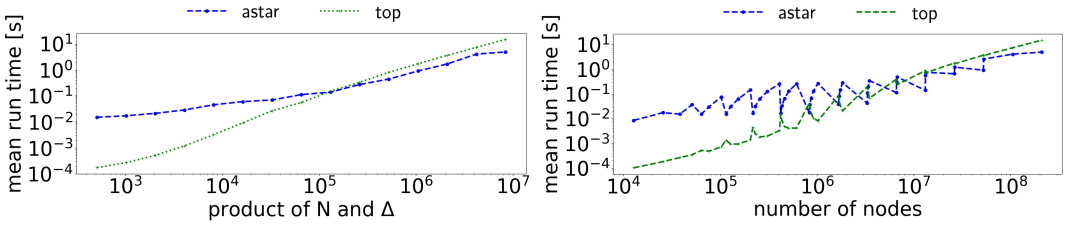

Figure 10.2: Left: Mean run times of (TOP) (green, + mark, dotted) and (Astar) (blue, circle mark, dashed) over the product of the discretization N and the trust-region radius Δ for (IP) instances stemming from (SH). Right: Mean run times of (TOP) (green, + mark, dotted) and (Astar) (blue, circle, dashed) over the number of nodes in the graphs for (IP) instances stemming from (SH). Original figure in [96].

We get a similar picture for the overall trend when considering the run times of (TOP) and (Astar) with respect to the number of nodes in the graph. However, the run times do not increase monotonically over the number of nodes in the graph and we observe a sequence of spikes in the run times, which is illustrated in Figure 10.2. We attribute the dominant spikes in the run times of (Astar) to the preprocessing step. The run time of the preprocessing step Algorithm 2 yields an offset for the run time of (Astar), which only depends on N and not on the current trust-region radius Δ.

We also observe a sequence of spikes in the overall run time trend of (TOP). The locations of these spikes seem to be opposed to spikes we observe for (Astar). We attribute these spikes to the fact that if N is relatively large compared to Δ, then the number of edges in the graph is relatively small compared to a graph with a similar value of $N\Delta$, where the ratio $\frac{N}{\Delta}$ is smaller.

Table 10.3: Cumulative run times in seconds of Algorithm 1 excluding the run times required for the subproblems of the form (IP) for all instances and start values on the different discretizations and parameterizations of (SH). Original table in [96].

α \ N	512	1024	2048	4096	8192	$\sum$
10^{-7}	439	1151	2147	3770	4056	11563
10^{-6}	450	585	456	429	1299	3219
10^{-5}	82	50	219	448	1066	1865
10^{-4}	99	137	288	313	1039	1876
10^{-3}	91	271	405	598	997	2362
$\sum$	1161	2194	3515	5558	8457	20885

Considering the time required for whole runs of Algorithm 1, we observe a substantial decrease when comparing the runs with (TOP) and (Astar) as subproblem solver to the same runs with (SCIP) as subproblem solver. The strongest effect can be observed for $N = 2048$, the largest case, where the necessary data is fully available. For $N = 2048$ we observe a decrease of the cumulative run time of the runs of Algorithm 1 from 30200 s with (SCIP) to 5242 s with (TOP) and 4083 s with (Astar). Thus, the subproblem solves consume 88.4 % of the run time of Algorithm 1 for (SCIP), 32.9 % for (TOP) an 13.9 % for (Astar). We have tabulated the cumulative run times of the runs of Algorithm 1 excluding the subproblem solvers over N and α in Table 10.3 and the cumulative run times of the subproblem solves over N and α for (Astar) and (TOP) in Table 10.4.

Table 10.4: Cumulative run times in seconds of (Astar) (top) and (TOP) (bot) for the subproblems of the form (IP) for all instances and start values on the different discretizations and parameterizations of (SH). Original table in [96].

α \ N	512	1024	2048	4096	8192	$\sum$
10^{-7}	32	97	313	2073	8258	10773
10^{-6}	29	52	72	279	2540	2972
10^{-5}	5	4	42	251	2378	2680
10^{-4}	6	12	58	216	1857	2149
10^{-3}	6	26	84	370	2019	2505
$\sum$	78	191	569	3189	17052	21079

α \ N	512	1024	2048	4096	8192	$\sum$
10^{-7}	29	201	1070	5614	18046	24960
10^{-6}	28	100	231	612	4859	5830
10^{-5}	5	7	98	637	4263	5010
10^{-4}	5	23	136	467	3949	4580
10^{-3}	5	41	190	836	4028	5100
$\sum$	72	372	1725	8166	35145	45480

Comparison and Hybridization with the Combinatorial Integral Approximation We execute the CIA (no switching constraints), see [61, 89], on the IOCP (SH) for the different discretizations, that is, we compute the solution of a continuous relaxation in a *first-discretize, then-optimize* manner, where $\alpha = 0$ and Ξ is replaced by $\operatorname{conv}\Xi = [-2, 23]$. Then we compute Ξ-valued controls by (a) using the initialization choices (1), (2), and (3) described in the beginning and (b) applying

sum-up rounding (no switching constraints or costs), see [88], to the solution of the continuous relaxation.

Table 10.5: Comparison of the mean of the relative difference between the objective values of the considered approaches to the solution of the relaxation of (SH) (computed with BFGS with $\alpha = 0$). SUR refers to the solution produced by sum-up rounding when initialized with the solution of the relaxation. SLIP(1) to SLIP(3) denote the runs of Algorithm 1 with the initial controls described in the beginning. SLIP(SUR) denotes the run of Algorithm 1 with the solution of sum-up rounding as initial control. Original table in [96].

N	α	SLIP(1)	SLIP(2)	SLIP(3)	SLIP(SUR)	SUR
512	1e-7	0.00023	0.11983	0.03849	**0.00022**	0.00048
	1e-6	**0.00169**	0.19134	0.11481	0.00198	0.00248
	1e-5	**0.00535**	4.49346	0.97448	0.00638	0.02242
	1e-4	0.05525	4.49346	1.00334	**0.05427**	0.22188
	1e-3	0.44930	4.49346	1.22963	**0.43289**	2.21646
1024	1e-7	**0.00032**	0.04969	0.00214	0.00049	0.00067
	1e-6	**0.00258**	0.51776	0.57327	0.00287	0.00455
	1e-5	0.01151	4.49713	1.21744	**0.00581**	0.04334
	1e-4	0.05737	4.49713	1.00014	**0.05484**	0.43123
	1e-3	0.43818	4.49713	1.22954	**0.42771**	4.31008
2048	1e-7	**0.00015**	0.02125	0.01961	0.00062	0.00198
	1e-6	0.01040	1.77102	0.97256	**0.00342**	0.00827
	1e-5	0.04375	4.50258	0.97502	**0.00952**	0.07120
	1e-4	**0.05492**	4.50258	1.00909	0.05744	0.70051
	1e-3	**0.43038**	4.50258	1.22905	0.43402	6.99362
4096	1e-7	0.00031	0.13618	0.07011	**0.00006**	0.00123
	1e-6	**0.00063**	4.45753	0.97230	0.00086	0.01034
	1e-5	0.00691	4.50187	0.97475	**0.00605**	0.10143
	1e-4	0.05730	4.50187	1.00366	**0.05473**	1.01231
	1e-3	**0.43380**	4.50187	1.23453	0.43383	10.12107
8192	1e-7	0.00036	0.40371	0.29677	**0.00017**	0.00162
	1e-6	**0.00069**	4.50151	0.98447	0.00100	0.01555
	1e-5	**0.00591**	4.50151	0.97527	**0.00591**	0.15487
	1e-4	**0.05489**	4.50151	0.99976	**0.05489**	1.54806
	1e-3	0.44950	4.50151	1.22991	**0.43848**	15.47995

The run times of the CIA are generally at least one order of magnitude lower, where the rounding steps (a) and (b) are negligible and thus the total run time amounts to the run time of BFGS. The objective values obtained with these two approaches are significantly lower than the objective values obtained with Algorithm 1

when the latter is initialized with zero. If Algorithm 1 is initialized with a solution to the CIA, which corresponds to a hybridization of both approaches, it is generally able to further reduce the gap to the continuous relaxation (up to an order of magnitude for fine discretizations and high values of α). The hybridization also yields substantially fewer iterations and thus a lower run time of Algorithm 1. We provide detailed computational results in Tables 10.5 and 10.6.

Table 10.6: Cumulative times for each approach executed on (SH). The runs are denoted as in Table 10.5. We note that by cumulative we mean that the run times of the solution of the continuous relaxation are included in the run times of SUR, SLIP(1), and SLIP(3) because their initializations use the solution of the continuous relaxation. Original table in [96].

N	α	SLIP(1)	SLIP(2)	SLIP(3)	SLIP(SUR)	SUR	BFGS
512	1e-7	14.88	286.18	203.07	12.53	11.07	11.07
	1e-6	18.74	296.58	196.56	14.62		
	1e-5	22.27	12.04	86.00	14.78		
	1e-4	24.88	11.69	102.38	13.98		
	1e-3	45.46	11.71	72.80	31.85		
1024	1e-7	43.78	752.11	556.33	40.51	34.73	34.73
	1e-6	39.24	415.32	285.98	74.96		
	1e-5	79.82	36.05	42.15	49.84		
	1e-4	64.66	35.74	153.18	44.92		
	1e-3	216.98	35.76	149.09	73.42		
2048	1e-7	202.06	1570.62	863.50	87.72	58.89	58.89
	1e-6	67.28	371.19	266.36	76.73		
	1e-5	67.79	60.48	309.33	162.20		
	1e-4	145.34	60.90	316.66	287.78		
	1e-3	367.91	60.49	237.39	154.42		
4096	1e-7	84.15	3798.13	2154.01	292.01	64.57	64.57
	1e-6	202.13	91.69	607.96	251.97		
	1e-5	291.71	66.88	534.04	273.39		
	1e-4	168.75	66.81	487.17	217.84		
	1e-3	658.59	66.89	436.54	392.61		
8192	1e-7	847.85	6471.02	5378.14	319.67	127.55	127.55
	1e-6	1887.68	134.06	2199.96	335.15		
	1e-5	1511.85	130.85	2183.99	622.64		
	1e-4	1458.45	130.92	1689.90	567.74		
	1e-3	1576.27	131.10	1691.36	1349.83		

We note that we have also tried to execute an adaptation of switching-cost aware rounding, see [12][1], on the solution of continuous relaxation to incorporate the total

[1]Free implementation `https://github.com/chrhansk/SCARP` downloaded on 11/01/2022.

variation penalty in the rounding of the continuous relaxation but the growth of the graph with respect to $|\Xi|$ (the best available estimate is exponential in $|\Xi|$, see (4.45) in [12]) lead to an infeasible consumption of RAM.

Remark 10.1. We note that the newest implementation of (TOP) allowed for a further decrease of the run time for (TOP). A major benefit of (TOP) is that it is straightforward to implement and interesting cases like boundary conditions can be included fast. Additionally, (TOP) has the advantage that not only the solution for a given trust-region radius Δ is calculated but for all positive integer values up to Δ. As the trust-region radius is reduced in the case that the step is rejected, this can eliminate most of the computational demand for these lower radii. Thus, the subroutines in the two-dimensional case use (TOP) instead of (Astar). Above in the numerical results, we have refrained from discussing a hybridization of (TOP) and (Astar) based on the trust-region radius as the advantages would be smaller now. Details can be found in [96].

10.1.2 Results for the two-dimensional case

To assess the performance of the primal heuristic, the branching priority and the cutting planes we introduce an advection-diffusion problem that serves as our benchmark problem. We run Algorithm 1 for a uniform square grid of size $N \times N$ and binary controls $\Xi = \{0, 1\}$ to produce subproblems of the form (TR-IP) for 5 different values for the parameter α. We compare 8 different combinations of the proposed tools for the time to reach optimality and the gap closed after a given time. Further details are given below. Moreover, we examine if the size of the fractional component in the root linear programming relaxation is an indicator for the hardness of the problem (TR-IP) and approximation quality of the Lagrangian relaxation.

The benchmark problem

Our advection-diffusion benchmark problem on $\Omega = (0, 1)^2$ reads

$$
\text{(AD)} \qquad \begin{aligned}
\min_{u,v} \quad & \frac{1}{2}||v - v_d||^2_{L^2(\Omega)} + \alpha \mathrm{TV}(u) \\
\text{s.t.} \quad & -\varepsilon \Delta v + c \cdot \nabla v = u \text{ in } \Omega, \\
& v|_{\Gamma_1} = 0, \\
& v|_{\Gamma_2}(x, y) = \sin(2\pi(x - 0.25)), \\
& \partial_n v|_{\Gamma_3} = 0 \\
& u(s) \in \{0, 1\}
\end{aligned}
$$

with $\varepsilon = 0.075$ and $c = (\cos(\pi/32), \sin(\pi/32)))^T$. For the boundary we define the subsets $\Gamma_1 = ([0, 0.25) \cup (0.75, 1]) \times \{0\} \cup \{0, 1\} \times (0, 1)$ and $\Gamma_2 = [0.25, 0.75] \times \{0\}$

for Dirichlet boundary conditions. The final subset is given by $\Gamma_3 = (0,1) \times 1$. v_d is the solution for the PDE with the control

$$u_d = \begin{cases} 3\exp(-\frac{1}{1-a(x,y)}) & \text{if } a(x,y) \leq 0.8 \\ 0 & \text{otherwise} \end{cases} + \begin{cases} 4\exp(-\frac{1}{1-b(x,y)}) & \text{if } b(x,y) \leq 0.4 \\ 0 & \text{otherwise} \end{cases}$$

where

$$a(x,y) = 20.25(x-0.25)^2 + 6.25(y-0.3)^2$$

and

$$b(x,y) = 2.25(x-0.8)^2 + 12.25(y-0.8)^2.$$

Implementation

We execute the SLIP algorithm on discretizations of the domain and PDE. We use the Python package FEniCSx, see [1, 7, 93, 94], for the discretization of the domain, the PDE, and the gradient computation, where we follow a *first-discretize, then-optimize* principle. We use a piecewise constant control ansatz on a uniform $N \times N$ grid of square cells, while the PDE is solved on a decomposition of the squares into 4 triangles and continuous Lagrange elements of order one are used.

We model the integer programs without the variables δ because they are not needed for the binary case as we can just use d as the absolute value and add signs in the remaining inequalties depending on the previous control value. We employ the integer programming solver Gurobi 10.0.0, see [51]. The subroutines (primal heuristic and calculation of the coefficients for the fully connected cutting planes) are implemented in C++ and we use pybind11, see [60], to call the implemented functions from Python. The branching priority is implemented via the branching priority parameter but note that instead all nodes in the even rows had the same high priority value which is a slight variation from Section 8.3.2. The cutting planes are added as lazy constraints to ensure that they are added to the model description. For a fair comparison the value *PreCrush* is set to 1, even when no cutting planes are added, because this setting significantly reduced the run time in our preliminary experiments.

Computational setup

We run the SLIP algorithm with the values $N = M \in \{32, 64\}$ and $\alpha \in \{4 \times 10^{-4}, 4\sqrt{5} \times 10^{-4}, 2 \times 10^{-3}, 2\sqrt{5} \times 10^{-3}, 1 \times 10^{-2}\}$ as well as $N = M = 96$ and $\alpha \in \{4\sqrt{5} \times 10^{-4}, 2 \times 10^{-3}, 2\sqrt{5} \times 10^{-3}\}$, where we chose only three values due to the computational demand. We set $\Delta^0 = \frac{1}{16}N^2$ and choose $\theta = 0.0001$ in Algorithm 1. To compare the approaches we solve each of the subproblems with the different combinations of tools that are detailed in Table 10.7. We set a time limit of 1 hour

for Gurobi before returning the current best primal point in the subproblem solver to generate instances of the form (TR-IP). We keep the default optimality tolerances of Gurobi which means a solution is considered optimal if the gap is less than 0.0001. We include the time to build the model in Gurobi in our measurements. Thus, the results include the whole run times of the subproblems but not the whole SLIP algorithm. The laptop for the experiments has an Intel(R) Core i7(TM) CPU with eight cores clocked at 2.5 GHz and 64 GB RAM.

Table 10.7: Abbreviations for the different solution approaches (combinations of the primal heuristic, the branching priority and the cutting planes). Original table in [74].

tools (Section)	none	p	b	c	p-b	p-c	b-c	p-b-c
Primal Heuristic (Section 8.3.1)	-	x	-	-	x	x	-	x
Branching Priority (Section 8.3.2)	-	-	x	-	x	-	x	x
Cutting Planes (Section 8.3.3)	-	-	-	x	-	x	x	x

Comparison of the computational results for the different approaches

We detail the number of instances of (IP) produced by Algorithm 1 in Table 10.8. The cumulative run times for all instances can be found in Table 10.9. The median run times are detailed in Table 10.10.

Table 10.8: Number of instances produced by the SLIP algorithm for the different values for N and α. Original table in [74].

N \ α	4×10^{-4}	$4\sqrt{5} \times 10^{-4}$	2×10^{-3}	$2\sqrt{5} \times 10^{-3}$	1×10^{-2}
32	40	34	24	15	1
64	70	38	53	19	1
96	-	74	173	35	-

For the smallest value $N = 32$ solving all 114 subproblems only takes around 2 to 3 minutes. The approaches *b-c* and *p-b-c* produce the best cumulative times with 115 and 116 seconds as seen in Table 10.9. On average it takes both approaches 1 second to solve an instance of this size. In general, the approaches using cutting planes (*c*, *p-c*, *b-c*, *p-b-c*) perform better compared to the alternatives which is also reflected in the median run times depicted in Table 10.10 where the approach *c* performs best. The approaches *b-c* and *p-b-c* perform best with a cumulative run time improvement of around 25 percent compared to the approach *none*.

Table 10.9: Cumulative run times of the different approaches for $N \in \{32, 64, 96\}$ and the 5 values for α. We note that *all* is the cumulative run time for all instances for the value of N. Original table in [74].

N	α	none	p	b	c	p-b	p-c	b-c	p-b-c
32	4×10^{-4}	**19**	20	20	22	21	22	22	22
	$4\sqrt{5} \times 10^{-4}$	23	23	23	**22**	24	23	23	23
	2×10^{-3}	35	36	36	**27**	37	**27**	**27**	**27**
	$2\sqrt{5} \times 10^{-3}$	48	47	40	33	39	31	**30**	**30**
	1×10^{-2}	30	31	23	17	23	17	**14**	**14**
	all	154	157	142	120	143	120	**115**	116
64	4×10^{-4}	2905	2932	2646	1492	2670	1470	971	**955**
	$4\sqrt{5} \times 10^{-4}$	1540	1537	1296	**594**	1289	606	645	706
	2×10^{-3}	7343	7437	5370	3625	5497	3531	2894	**2876**
	$2\sqrt{5} \times 10^{-3}$	16643	14554	8586	9326	8822	8874	6624	**6448**
	1×10^{-2}	**3600**	3601	3601	3602	3601	3601	3601	3601
	all	32032	30061	21498	18639	21878	18082	14736	**14586**
96	$4\sqrt{5} \times 10^{-4}$	61292	-	-	-	-	-	-	**46382**
	2×10^{-3}	383256	-	-	-	-	-	-	**350115**
	$2\sqrt{5} \times 10^{-3}$	78743	-	-	-	-	-	-	**65587**
	all	523292	-	-	-	-	-	-	**462083**

Table 10.10: Median run times of the different approaches for $N \in \{32, 64, 96\}$ and the 5 values for α. We note that *all* is the median run time for all instances for the value of N. Original table in [74].

N	α	none	p	b	c	p-b	p-c	b-c	p-b-c
32	4×10^{-4}	**0.31**	0.32	0.33	0.43	0.34	0.50	0.44	0.47
	$4\sqrt{5} \times 10^{-4}$	0.68	0.63	0.66	**0.61**	**0.61**	0.64	0.67	0.69
	2×10^{-3}	1.28	1.24	1.29	1.00	1.32	1.00	**0.89**	0.90
	$2\sqrt{5} \times 10^{-3}$	2.71	2.64	2.35	1.67	2.33	1.66	**1.57**	1.62
	1×10^{-2}	29.71	30.72	22.55	16.83	22.59	16.80	13.79	**13.76**
	all	0.74	0.77	0.78	**0.67**	0.83	0.69	0.71	0.72
64	4×10^{-4}	12.47	13.38	12.11	11.32	12.18	11.42	**10.89**	11.00
	$4\sqrt{5} \times 10^{-4}$	10.98	11.43	10.26	9.23	10.35	8.66	8.95	**8.44**
	2×10^{-3}	82.84	83.35	61.40	37.48	63.12	37.40	29.83	**29.66**
	$2\sqrt{5} \times 10^{-3}$	153.04	152.87	137.63	83.62	137.80	**83.58**	86.41	86.38
	1×10^{-2}	**3600.46**	3600.57	3600.55	3602.37	3600.54	3601.04	3601.15	3601.15
	all	26.78	26.88	23.00	16.95	23.77	16.33	15.02	**14.40**
96	$4\sqrt{5} \times 10^{-4}$	298.61	-	-	-	-	-	-	**121.39**
	2×10^{-3}	2617.74	-	-	-	-	-	-	**2433.26**
	$2\sqrt{5} \times 10^{-3}$	3600.93	-	-	-	-	-	-	**1705.99**
	all	1579.09	-	-	-	-	-	-	**989.70**

This behaviour extends to the case with $N = 64$ with an improvement of 46 percent. For this discretization most instances are solved within 600 seconds for the different approaches. We note that the approaches using the cutting planes (*c*, *p*-*c*, *b*-*c*, *p*-*b*-*c*) perform better than those which do not. In general, the approaches which include the primal heuristic (*p*, *p*-*b*, *p*-*c*, *p*-*b*-*c*) only seem to improve the run time if combined with the cutting planes (*p*-*c*, *p*-*b*-*c*). We note that the improvements of *p*-*b*-*c* compared to *b*-*c* are small which indicates that for a large non-binary set Ξ the primal heuristic might even increase the runtime as the time complexity of underlying shortest path approach is quadratic in $|\Xi|$. Both observations are illustrated by the performance plots in Figure 10.3.

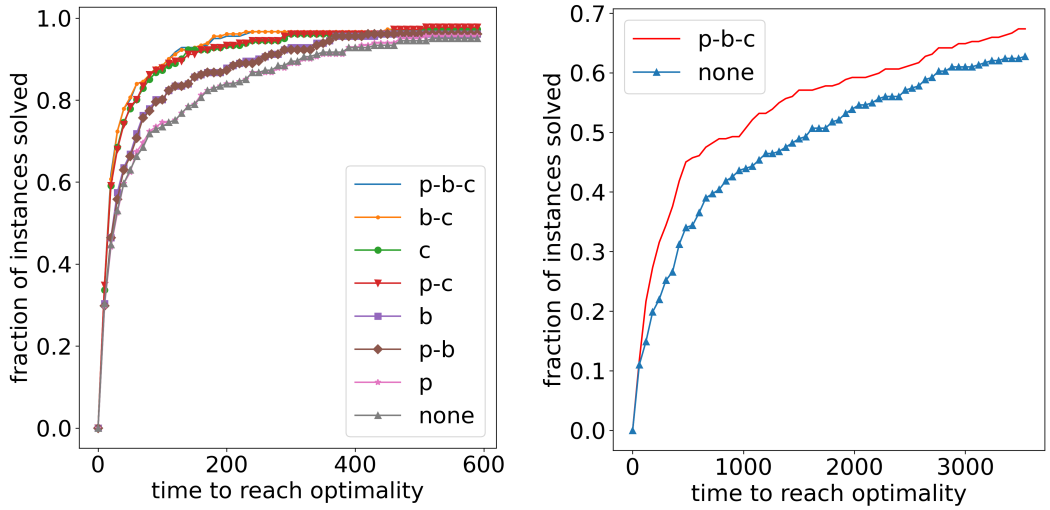

Figure 10.3: Performance plots for $N = 64$ (left) and $N = 96$ (right). The plots visualize what fraction of the instances are solved after a given time. For $N = 64$ nearly all instances are solved after 600 seconds while for $N = 96$ a significant number of instances are not solved within the time limit of 1 hour. Original figure in [74].

The best approach for both $N = 32$ and $N = 64$ is the combination of all the tools as the cumulative run times are lowest or second lowest, see Table 10.9. For $N = 64$ and $\alpha = 1 \times 10^{-2}$ the instance could not be solved within any time limit. All approaches produce the same primal point with objective value 0. The objective lower bounds produced vary from -0.18 by *none* to -0.08 by *p*-*b*-*c*.

For $N = 96$ we only compare the approaches with no tools and all tools combined. An improvement of 12 percent for the cumulative run time is achieved by employing all proposed tools. We note that a larger improvement of 37 percent is achieved regarding the median run times. This effect might be due to the significant number of instances reaching the time limit of 1 hour which is visualized by the performance plot in Figure 10.3. There were 68 instances which could not be solved within the time limit by either approach. Additionally, 23 instances could not be solved by *p*-*b*-*c*

but could be solved by *none*, while the opposite case occurred for 37 instances. The mean duality gap is reduced from 2.6 percent to 2.3 percent by *p-b-c*. Furthermore, the highest occurring duality gap is also reduced as visualized by the violin plot in Figure 10.4. The primal values were significantly better (exceeding the tolerance of the integer programming solver) for the approach "none" in 15 cases while *p-b-c* produced significantly better primal values in 23 cases.

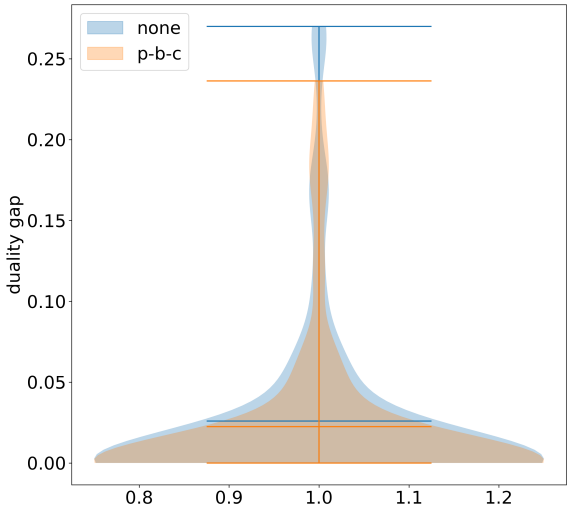

Figure 10.4: The violin plot shows the distribution of the remaining duality gap after the time limit of 1 hour. The lines in the middle mark the mean duality gaps for all instances with $N = 96$. Original figure in [74].

We reran the experiments with a time limit of 3 hours for all instances only solved by either *p-b-c* or *none* to get a clearer picture. We see that now the new cumulative run times of the approaches are 530781 seconds for *p-b-c* and 655740 seconds for *none*. Thus, *p-b-c* improves the run time by 19 percent compared to *none*. We note that 3 instances for *p-b-c* still reached the new time limit while 7 instances reached the new time limit for *none*.

In many cases the feasible point obtained from the Lagrangian relaxation did not produce an approximation as no capacity was used. However, for $N = 32$ there were 21 instances for which at least half of the capacity was used by the feasible point. This effect gets smaller as N increases as for $N = 64$ there were 16 such instances while for $N = 96$ only 5 instances produced such a feasible point from the Lagrangian relaxation. In these cases the approximation guarantee from Theorem 4.4 was always achieved. In five cases the feasible point was optimal and in the remaining cases the feasible point was strictly better than the approximation guarantee derived in Theorem 4.4.

In Figure 10.5 we see that a larger fractional component in the root linear program corresponds to a higher run time. We note that for $N = 96$ the linear regression is

negatively impacted by the large amount of instances reaching the time limit and other regressions may be a better fit for the data but still shows the general trend.

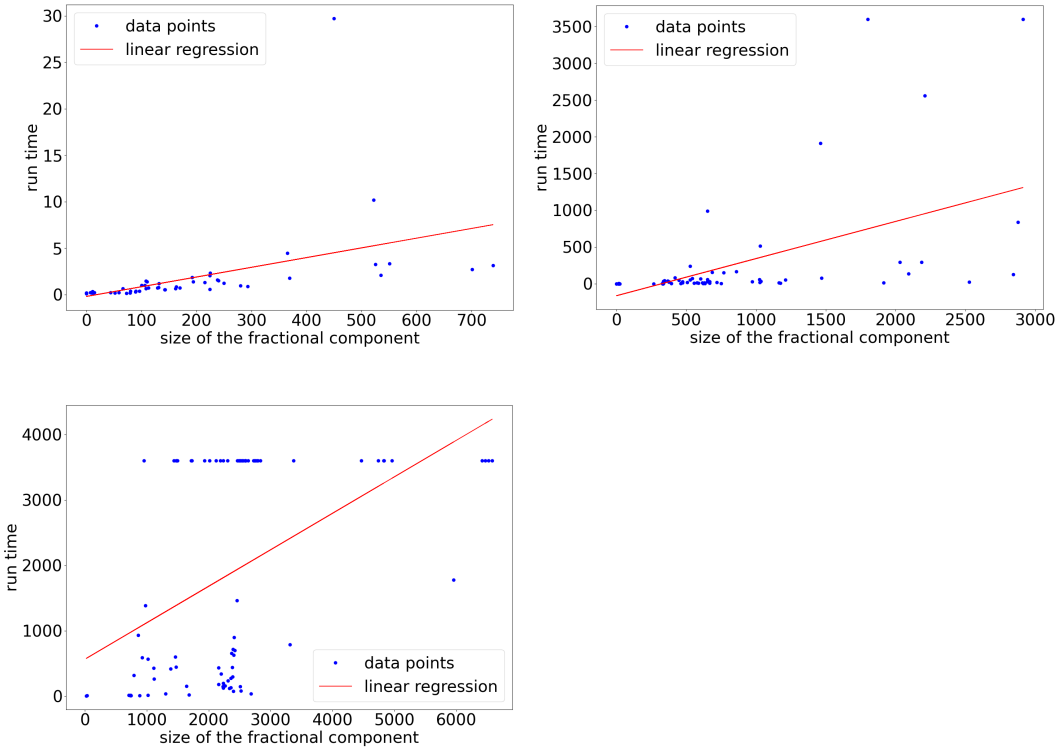

Figure 10.5: From left to right, top to bottom, the plots show the relation between the size of the fractional component and the run time (none) for $N = 32$ and $\Delta = 64$, $N = 64$ and $\Delta = 256$, $N = 96$ and $\Delta = 576$. In each plot the red line represents the linear regression of the data which itself is plotted as the blue points. Original figure in [74], newly created from data.

10.2 Numerical results for the overall trust-region algorithm

In this section we provide unpublished numerical results which not only focus on the individual (TR-IP) but the whole trust-region algorithm. In Section 10.2.1 we use the 2D test problem presented in [6] to validate the newly proposed decompositions by comparing them to the decompositions used in [6]. Furthermore, we test the different decompositions without and in combination with the primal heuristic, branching priority and cutting planes. In Section 10.2.2 we provide an imaging problem with a non-binary control set to test if the extension of the cutting planes to the non-binary case in combination with the branching priority prove helpful. When it is stated that

cutting planes are used this means that both types described in Section 8.3.3 are implemented.

10.2.1 The decomposition approach applied to an advection-diffusion problem

We have already used an advection-diffusion problem to validate the primal heuristic, the branching priority and the cutting planes (*p-b-c*) in the previous section. The 2D test case from [6] is also a advection-diffusion equation with a binary control set Ξ. We run the Algorithm 1 without a decomposition as well as Algorithm 4 for 8 different decompositions of the domain. We use uniform discretization grids and thus the original domain is discretized into a $N \times N$ square grid. We compare 3 different values for N and 2 values for α. The algorithms are run once with the primal heuristic, the branching priority and cutting planes (*p-b-c*) and once without. Further details are given in the following.

The benchmark problem

As stated, the set Ξ is binary and we use the example from [6]. The control is denoted by u and the state vector v is the solution to

$$\text{(CD)} \qquad \begin{aligned} &-\varepsilon\Delta v + c_1 \cdot \nabla v + 2vu = f \text{ in } \Omega, \\ &v|_{\Gamma_1} = 0, \\ &v|_{\Gamma_2} = \sin(2\pi(x_1 - 0.25)), \\ &\partial_n v|_{\Gamma_3} = 0, \end{aligned}$$

where $c_1(x) = (\sin(\pi x_1)\ \cos(2\pi x_2))^T$ for $x \in \Omega$, $f(x) = \sin(2\pi x_1 + 2\pi x_2) + 3$ for $x \in \Omega$, and $\varepsilon = 0.04$. For the boundary we define the subsets $\Gamma_1 = ((0, 0.25) \cup (0.75, 1)) \times \{0\} \cup \{0, 1\} \times (0, 1)$ and $\Gamma_2 = (0.25, 0.75) \times \{0\}$ for Dirichlet boundary conditions. The final subset is given by $\Gamma_3 = (0, 1) \times \{1\}$. The objective function F is given by $F = j \circ S$ with $j(v) = \frac{1}{2}||v - v_d||^2_{L^2}$, where S is the solution operator to (CD) and v_d is the solution to the problem (CD) with a changed constant $\tilde{c}_1 = (-x_2\ 2x_1)^T$ instead of c_1 and a control $u_d = 2.5\chi_A - 4(x_1 - 0.35)^3\chi_A - 6(x_2 - 0.35)^3\chi_B$ with $A = (0, 0.35)^2$ and $B = \Omega \setminus (0, 0.35)^2$. u_d is choosen in a way that it can not be attained by u, see [6].

Implementation

We follow the setup from [6] on the PDE side. For the discretization and the finite element approach the package FEniCSx [7] is used with a piecewise constant control ansatz on a uniform $N \times N$ grid of square cells. The PDE is solved on a decomposition of each square into 4 triangles and continuous Lagrange elements of order

one are used for v, v_d. The optimization follows the *first-discretize, then-optimize* principle such that for $\nabla F(u)$ the adjoint and the finite element system are calculated for the previously fixed grid. We use the integer programming solver Gurobi 11.0.0 [51] to solve the integer programs arising as subproblems. The subroutines are implemented in C++ and we use pybind11, see [60], to call the implemented functions from Python. The branching priority is implemented via the branching priority parameter. The cutting planes are implemented as cuts, not lazy constraints. The setting PreCrush is set to 1.

Computational setup

We choose the discretizations $N \in \{64, 96, 128\}$ and the two α values $\{10^{-3}, 1.25 \cdot 10^{-3}\}$ which proved most significant in [6] with respect to the run times. Furthermore, we choose $\Delta^0 = 0.125N^2$ and $\theta = 10^{-4}$ for the algorithmic setup of the trust-region approaches. We start with $u^0 \equiv 0$. We run Algorithm 1 once without and once with *p-b-c* to determine the run times without decompositions (No Patch). We choose the first integer program for (No Patch) with a run time of more than 10 seconds for $N = 64$ and 60 seconds for $N \in \{96, 128\}$ from a preliminary run to determine the decomposition based on the fractional component of the root linear programming solution. We determine the middle point of the fractional component for each of the root linear programs and decompose the domain into 4 patches split by the horizontal and vertical line meeting in the middle point with some additional overlap, see Chapter 9 regarding the need for overlap. We have visualized the idea to obtain the middle point in Figure 10.6.

We run Algorithm 4 for the decompositions

- into 4, 9, 16 squares (SQR4, SQR9, SQR16),
- into 4, 9 horizontal stripes (HOR4, HOR9),
- into 4, 9 vertical stripes (VER4, VER9),
- into 4 patches based on the fractional component (FRAC4).

We set no time limit for Gurobi and do not change the standard Gurobi tolerances. We set a time limit for each run of the trust-region algorithms of 10 days and run the experiments on a single node of the LiDO3, a HPC605 Linux cluster at TU Dortmund University with AMD EPYX 7542 32-Core CPUs and 64 GB RAM. (Computation restricted to one CPU.)

Numerical results

All results are given in Table 10.11. The objective values for all decompositions barely differ. The biggest difference in objective values for $N =$

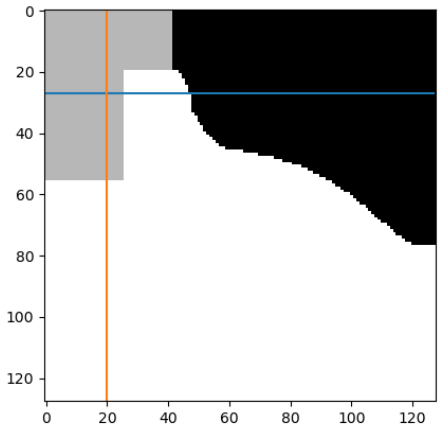

Figure 10.6: Given is a heat map of the values of a linear programming relaxation on the domain. The fractional component is marked in grey while black represents the value 0 and white represents the value 1. We split the domain at the intersection of the orange and blue line where each line splits the bounding box around the fractional component in the middle along one dimension. We note that we have added the needed additional overlap afterwards.

128 can be seen for $\alpha = 1.25 \cdot 10^{-3}$ for (VER9) with 0.674281 and (No Patch)/(SQR4)/(HOR4)/(VER4)/(HOR9)/(FRAC4) with 0.674099. Furthermore, we see the that *p-b-c* does not change the objective value (in theory this could very well happen if there exist two optimal points for an IP with regards to the standard Gurobi tolerance and the trust-region algorithm thus does different steps depending on the choice).

We first discuss the run time results for the different decompositions without *p-b-c*. For $\alpha = 10^{-3}$ and $N = 64$ all square decompositions manage to decrease the run times. We also see a trend that more patches result in a decrease in run time from 264.65 seconds to 201.84/88.13/43.83 seconds respectively for (SQR4)/(SQR9)/(SQR16). While 4 patches reduce the run time by around 24 percent, this value is increased to 66 percent for 9 and to 83 percent for 16 squares. For $\alpha = 10^{-3}$ and $N = 96$ this observation also holds true. We see a decrease of 89-95 percent from 6783.70 seconds to 721.46/459.45/326.22 for (SQR4)/(SQR9)/(SQR16). For $\alpha = 10^{-3}$ and $N = 128$ we see a shift in behaviour as using only 4 square patches increases the run time significantly from 113229.66 to 345874.03 seconds. For 9 and 16 square patches we still see a run time decrease by 99 percent and 99.7 percent with 1354.98 and 369.82 respectively. For $\alpha = 1.25 \cdot 10^{-3}$ we see the same trends, however, using only 4 square patches is worse for every value of N. Especially in the case $N = 128$, we see a sharp increase in computational demand and the algorithm did not terminate in 10 days (the time limit for a job on the compute infrastructure).

Table 10.11: Table of the objective values and run times of Algorithm 1 and Algorithm 4 for different decompositions with and without the primal heuristic, branching priority and cutting planes *p-b-c* which is denoted by p-b-c and base respectively. The best time between the two is highlighted.

N	Patches	α (in 10^{-3})	J(p-b-c)	J(base)	f(p-b-c)	tv(p-b-c)	f(base)	tv(base)	t(p-b-c)	t(base)
64	No Patch	1	0.675851	0.675851	0.674569	1.281250	0.674569	1.281250	**225.37**	264.65
		1.25	0.676169	0.676169	0.674587	1.265625	0.674587	1.265625	**128.85**	174.86
	SQR4	1	0.675851	0.675851	0.674569	1.28125	0.674569	1.28125	**146.74**	201.84
		1.25	0.676169	0.676169	0.674587	1.265625	0.674587	1.265625	**169.51**	282.13
	SQR9	1	0.675842	0.675842	0.674123	1.71875	0.674123	1.71875	105.10	**88.13**
		1.25	0.676269	0.676269	0.674179	1.671875	0.674179	1.671875	132.39	**107.59**
	SQR16	1	0.675836	0.675836	0.674086	1.75	0.674086	1.75	56.24	**43.83**
		1.25	0.676273	0.676273	0.674086	1.75	0.674086	1.75	54.93	**42.87**
	HOR4	1	0.675846	0.675846	0.674065	1.78125	0.674065	1.78125	77.57	**61.65**
		1.25	0.676169	0.676169	0.674587	1.265625	0.674587	1.265625	50.17	**35.86**
	VER4	1	0.675851	0.675851	0.674569	1.28125	0.674569	1.28125	108.59	**106.89**
		1.25	0.676169	0.676169	0.674587	1.265625	0.674587	1.265625	94.46	**79.27**
	HOR9	1	0.675851	0.675851	0.674569	1.28125	0.674569	1.28125	42.96	**28.07**
		1.25	0.676169	0.676169	0.674587	1.265625	0.674587	1.265625	42.74	**28.22**
	VER9	1	0.675848	0.675848	0.674098	1.75	0.674098	1.75	45.67	**29.36**
		1.25	0.676338	0.676338	0.674542	1.4375	0.674542	1.4375	49.60	**32.81**
	FRAC4	1	0.675851	0.675851	0.674569	1.28125	0.674569	1.28125	104.53	**99.30**
		1.25	0.676169	0.676169	0.674587	1.265625	0.674587	1.265625	73.67	**56.33**
96	No Patch	1	0.672009	0.672009	0.670738	1.270833	0.670738	1.270833	**1447.17**	6783.70
		1.25	0.672325	0.672325	0.670750	1.260417	0.670750	1.260417	**2093.96**	22294.37
	SQR4	1	0.672041	0.672041	0.670301	1.739583	0.670301	1.739583	938.48	**721.46**
		1.25	0.672325	0.672325	0.67075	1.260417	0.67075	1.260417	**1269.99**	39388.89
	SQR9	1	0.672041	0.672041	0.670301	1.739583	0.670301	1.739583	554.54	**459.45**
		1.25	0.672325	0.672325	0.67075	1.260417	0.67075	1.260417	**1079.20**	13032.72
	SQR16	1	0.672048	0.672048	0.670266	1.78125	0.670266	1.78125	372.61	**326.22**
		1.25	0.672475	0.672475	0.670366	1.6875	0.670366	1.6875	430.04	**378.13**
	HOR4	1	0.672009	0.672009	0.670738	1.270833	0.670738	1.270833	**310.43**	312.60
		1.25	0.672325	0.672325	0.67075	1.260417	0.67075	1.260417	**221.52**	239.04
	VER4	1	0.672009	0.672009	0.670738	1.270833	0.670738	1.270833	**439.61**	605.86
		1.25	0.672325	0.672325	0.67075	1.260417	0.67075	1.260417	**334.11**	375.78
	HOR9	1	0.672009	0.672009	0.670738	1.270833	0.670738	1.270833	157.20	**109.05**
		1.25	0.672325	0.672325	0.67075	1.260417	0.67075	1.260417	132.25	**86.13**
	VER9	1	0.672048	0.672048	0.670308	1.739583	0.670308	1.739583	197.87	**134.86**
		1.25	0.672332	0.672332	0.67073	1.28125	0.67073	1.28125	197.24	**137.38**
	FRAC4	1	0.672041	0.672041	0.670302	1.739583	0.670302	1.739583	551.59	**418.36**
		1.25	0.672325	0.672325	0.67075	1.260417	0.67075	1.260417	**686.78**	999.82
128	No Patch	1	0.673781	0.673781	0.672508	1.273438	0.672508	1.273438	**9734.88**	113229.66
		1.25	0.674099	0.674099	0.672517	1.265625	0.672517	1.265625	**11754.57**	657389.48
	SQR4	1	0.673781	0.673781	0.672508	1.273438	0.672508	1.273438	**12840.60**	345874.03
		1.25	0.674099	-	0.672517	1.265625	-	-	**11047.18**	>10 days
	SQR9	1	0.673787	0.673787	0.672053	1.734375	0.672053	1.734375	**1337.34**	1354.98
		1.25	0.674217	0.674217	0.672098	1.695312	0.672098	1.695312	1325.69	**1121.54**
	SQR16	1	0.673792	0.673792	0.672026	1.765625	0.672026	1.765625	475.77	**369.82**
		1.25	0.674218	0.674218	0.672099	1.695312	0.672099	1.695312	641.31	**495.66**
	HOR4	1	0.673781	0.673781	0.672508	1.273438	0.672508	1.273438	**942.62**	1379.53
		1.25	0.674099	0.674099	0.672517	1.265625	0.672517	1.265625	**600.61**	2461.56
	VER4	1	0.673781	0.673781	0.672508	1.273438	0.672508	1.273438	**1763.11**	11055.09
		1.25	0.674099	0.674099	0.672517	1.265625	0.672517	1.265625	**1767.64**	3678.27
	HOR9	1	0.673781	0.673781	0.672508	1.273438	0.672508	1.273438	300.18	**219.30**
		1.25	0.674099	0.674099	0.672517	1.265625	0.672517	1.265625	291.61	**210.57**
	VER9	1	0.673789	0.673789	0.672055	1.734375	0.672055	1.734375	387.20	**293.80**
		1.25	0.674281	0.674281	0.672484	1.4375	0.672484	1.4375	381.85	**289.82**
	FRAC4	1	0.673781	0.673781	0.672508	1.273438	0.672508	1.273438	975.69	**890.90**
		1.25	0.674099	0.674099	0.672517	1.265625	0.672517	1.265625	441.10	**329.61**

In comparison to the square case our newly proposed decompositions always manage to decrease the run times significantly, no matter the value for α and N. The decompositon into stripes with 4 patches are slightly worse than (SQR9) for larger values of N but clearly outperform the squares approach when using 9 stripes. Both manage to outperform (SQR16) for $N = 128$ for both values of α. The best performance over all decompositions is achieved by using 9 horizontal patches where we see a decrease from 113229.66 seconds to 219.30 seconds for $N = 128$ and $\alpha = 10^{-3}$ and from 657389.48 seconds to 210.57 seconds for $N = 128$ and $\alpha = 1.25 \cdot 10^{-3}$. Our decomposition based on the fractional component outperforms every other decomposition with the same number of patches, meaning 4 patches. The run times for this decomposition are able to outperform (SQR9) for every combination of N and α. For $N = 128$ it takes 890.90 seconds for $\alpha = 10^{-3}$ and 329.61 seconds for $\alpha = 1.25 \cdot 10^{-3}$ with this decomposition.

When the decompositions are combined with *p-b-c* we see a general trend that *p-b-c* improves the run time when the run times were high without *p-b-c* but has a negative impact in the cases that the decomposition itself managed to decrease the run times significantly. We observe that for $N = 64$ *p-b-c* only improves the run times for (No Patch) and (SQR4) while the run times for the remaining decompositions actually increase if *p-b-c* is used. We do, however, observe that as we increase N this behaviour changes. For $N = 128$ we see that using *p-b-c* is now advantages for (HOR4) and (VER4) as well. For $\alpha = 10^{-3}$ *p-b-c* also performs better for (SQR9). We see significant decreases for (No Patch) from 113229.66 seconds to 9734.88 seconds for $\alpha = 10^{-3}$ and from 657389.48 seconds to 11754.57 seconds for $\alpha = 1.25 \cdot 10^{-3}$. The same holds true for (SQR4) where the negative effect of the decomposition is mostly negated by *p-b-c* such that this decomposition performs similar to (No Patch) in the case that *p-b-c* is used for both.

Interpretation of the numerical results

As in [6] the different decompositions return objective values very close to the objective value returned by the trust-region algorithm without the decomposition. The differences are so small that they do not produce a difference in the quality of the solutions found between the decompositions. This also holds true for the newly proposed decompositions. The results regarding the objective values thus do not give any reasons against the use of any of the decompositions at all. We do not observe a trade-off between run times and objective values.

We have seen that the newly introduced decompositions manage to outperform the decompositions into squares. Surprisingly, the decomposition into 9 horizontal stripes (HOR9) clearly outperforms (SQR16), (FRAC4) and the decomposition into 9 vertical stripes (VER9), most notably for $N = 128$. In [6] the trend was clear that more patches equal lower run times which is not the case here. We hypothesize that

the especially low run times for (HOR9) are due to the same effect as for (FRAC4). The horizontal stripes manage to split the fractional components on which (FRAC4) is based for the different values of N and α very well as the larger sides of these fractional components were in the vertical direction. The flip side of this effect is seen in the case of (SQR4). The fractional components occurring for (No Patch), on which (FRAC4) is based on, seem to often lie completely/mostly in one of the 4 patches of (SQR4). The integer programs of this patch are responsible for the high run times of (SQR4) as for example 99 percent of the run time of the integer programs occurs in the top-left patch for $N = 128$ and $\alpha = 10^{-3}$. We conjecture that the problems get easier when a decomposition is used which manages to split the domain in such a way, that the fractional components which occur in the root linear programs during the run of (No Patch) do not lie in a single patch but are distributed more evenly between multiple patches. On the other hand, the problems get harder when the fractional component lies completely in one patch. *p-b-c* seems mostly useful when the integer programs are not solved instantly, otherwise the additional computational demand is not offset by a reduction of the run time due to a smaller size of the branching tree.

10.2.2 The decomposition approach applied to an imaging problem

We use an imaging denoising problem derived in the numerical experiments in [91]. The control value set is now non-binary and given by $\Xi = \{0, \dots, 5\}$ to model the greyscale. We run Algorithm 1 and Algorithm 4 with 4 different decompositions which are a subset of the decompositions presented for the previous numerical example. We use an $N \times N$ uniform discretization grid. We choose 2 values of N and a single α value. We run the experiments once with the branching priority and the cutting planes (*b-c*) as well as once without. We note that we chose not to include the primal heuristic as the time complexity of the underlying shortest path computation is quadratic in Ξ and might increase the run time instead of decreasing it.

The benchmark problem

We concern ourselves with the problem

$$\text{(P-IMG)} \qquad \begin{aligned} \min_{u \in L^2(\Omega)} \quad & \frac{1}{2}\|y - y_d\|^2_{L^2(\Omega)} + \tilde{\alpha}\mathrm{TV}(c_s u) \\ \text{s.t.} \quad & -\kappa\Delta y + y = c_s u \text{ in } \Omega, \quad \partial_n y = 0 \text{ on } \partial\Omega \\ & u(x) \in U = \{0, \dots, 5\} \subset \mathbb{Z} \text{ for a.a. } x \in \Omega = (0,1)^2 \end{aligned}$$

where $\kappa = 5{\cdot}10^{-4}$, $c_s = \frac{1}{5}$ is a constant for scaling to 6 control values to the range [0,1] (our reference greyscale range), $\tilde{\alpha} = \frac{1}{c_s}\alpha$ and we have Neumann boundary conditions.

To obtain the form (IOCP), we use the linear solution operator $S_N : L^2(\Omega) \to H^1(\Omega)$ for the PDE

$$-\kappa\Delta y + y = c_s u \text{ in } \Omega, \quad \partial_n y = 0 \text{ on } \partial\Omega,$$

to map $u \in L^2(\Omega)$ to the solution $y \in H^1(\Omega)$, see [91] for details regarding the fulfillment of the assumptions for the trust-region algorithm. We determine y_d by applying S_N to u_d where the latter is calculated as the greyscaled image to which Gaussian noise with a mean of 0 and standard deviation of 0.025 is added.

We hope to observe two effects from the above problems:

- We expect to obtain a solution which does contain all (or most) control values in different areas of the underlying discretization grid and not just the smallest and largest values. We have discussed the role of the terms $M(\Delta_{out} - \sum_{(i,j)\in H} \delta_{i,j})$ and $M(\Delta_{out_B} - \sum_{(i,j)\in H_B} \delta_{i,j})$ of the cutting planes in Section 8.3. We have argued that this is different from the original construction if the previous control values are not the minimal or maximal values in Ξ which potentially weakens the cutting plane. Thus, the described behaviour is of interest to observe how well the proposed extension of the cutting planes in combination with the branch priority work in practice.

- We should clearly see the anisotropic behaviour described in Remark 3.7 in the final result. This shows the need for the ideas in [92] and [91]. It is desirable to include the ideas from this dissertation into the framework of [92] and [91] in future research.

Implementation

We again use the same implementation description as for the previous numerical example but have switched off the primal heuristic.

Computational setup

The values for N are given by $N \in \{96, 128\}$ and we set $\alpha = 2 \cdot 10^{-4}$ such that $\tilde{\alpha} = 10^{-3}$. The trust-region reset radius is $\Delta^0 = 0.125N^2$ and we choose $\theta = 10^{-4}$. We start with $u^0 \equiv 0$. We run Algorithm 1 without a decomposition (No Patch) once with *b-c* and once without. We run Algorithm 4 for the decomposition into 9 squares (SQR9), 9 horizontal stripes (HOR9), 9 vertical stripes (VER9) and 4 patches based on the fractional component (FRAC4), each with and without *b-c*. We set no time limit for Gurobi and do not change the standard Gurobi tolerances. We set a time limit for each run of the trust-region algorithms of 10 days and run the experiments on a single node of the LiDO3, a HPC605 Linux cluster at TU Dortmund University with AMD EPYX 7542 32-Core CPUs and 64 GB RAM. (Computation restricted to one CPU.)

Table 10.12: Table of the objective values and run times of Algorithm 1 and Algorithm 4 for different decompositions with and without the branching priority and cutting planes *b-c* which is denoted by b-c and base respectively. We have highlighted the best objective value and the best run time in every row.

N	Patches	α	J(b-c)	J(base)	f(b-c)	tv(b-c)	f(base)	tv(base)	t(b-c)	t(base)
96	No Patch	0.0002	0.004199	-	0.000987	16.062500	-	-	**391326.15**	>25 days
	SQR9	0.0002	**0.004194**	0.004199	0.000965	16.145833	0.000984	16.072917	33278.89	**29522.99**
	HOR9	0.0002	**0.004217**	0.004218	0.000954	16.312500	0.000941	16.385417	9621.24	**9164.48**
	VER9	0.0002	0.004220	**0.004219**	0.000956	16.322917	0.000955	16.322917	6665.97	**6337.65**
	FRAC4	0.0002	0.004208	**0.004206**	0.000968	16.197917	0.000972	16.166667	**124944.98**	173955.01
128	No Patch	0.0002	-	-	-	-	-	-	>18 days	>18 days
	SQR9	0.0002	0.004197	**0.004196**	0.001015	15.906250	0.001009	15.937500	**117331.12**	386220.41
	HOR9	0.0002	**0.004227**	0.004233	0.000952	16.375000	0.000958	16.375000	**33931.82**	41097.60
	VER9	0.0002	0.004222	**0.004220**	0.000968	16.273438	0.000960	16.296875	**28105.39**	31085.11
	FRAC4	0.0002	-	-	-	-	-	-	>18 days	-

Numerical results

The numerical results show similarities but also differences compared to the previous numerical experiments in Section 10.2.1. We have tabulated the objective values and run times in Table 10.12. We note for $N = 128$ that (No Patch) with and without *b-c* did not finish even after extending the time limit to 18 days. Without *b-c* 106 outer iterations of the trust-region algorithm were achieved while 245 outer iterations were achieved with *b-c* (note that the final number of iterations does not have to be identical but is typically close). For (FRAC4) with *b-c* we also run into the extended time limit. We decided against running (FRAC4) without *b-c* afterwards because for $N = 96$ the run time was already significantly reduced by *b-c* and we do not expect (FRAC4) without *b-c* to finish in the time limit. For $N = 96$ (No Patch) without *b-c* did not finish in the extended time limit of 25 days.

We discuss the run times and start with $N = 96$. We see that (No Patch) clearly performs the worst with and without *b-c* (391326.15 seconds and above the extended time limit of 25 days respectively). Compared to our previous experiment, the decomposition into 4 patches does outperform (No Patch) significantly with 124944.98 and 173955.01 seconds respectively, but is clearly slower than any decomposition into 9 patches. Between the decompositions with 9 patches, we see clearly that the decomposition into squares (SQR9) performed worse than the decompositions into 9 stripes (HOR9, VER9) with respect to the run times. It took (SQR9) 33278.89 seconds with *b-c* and 29522.99 seconds without *b-c* while (HOR9) finished in 9621.24 seconds with *b-c* and 9164.48 seconds without *b-c*. The best performance was achieved by (VER9) with 6665.97 and 6337.65 seconds respectively with and without *b-c*.

For $N = 128$ we see that the some trends hold. (No Patch) is not able to finish in 18 days with and without *b-c* while all decompositions with 9 patches are able to. Between the decompositions with 9 patches (SQR9) performs the worst with 117331.12 seconds with *b-c* and 386220.41 seconds without. The second best performing decomposition is (HOR9) with 33931.82 seconds with *b-c* and 41097.60 seconds without

Table 10.13: The table contains the mean, median and highest run time of the integer programs occurring for each decomposition. Furthermore we give the mean and median duality gap *in percent*. As most instances have a gap of 0 we have denoted by (greater 0) that we only considered instances with positive gaps. Note that the values of (No Patch) for $N = 128$ are only using instances calculated in the 18 days before termination. The same holds for (No Patch) without *b-c* for $N = 96$ with a time limit of 25 days.

N	Patches	mean time	median time	highest time	mean gap	mean gap (greater 0)	median gap (greater 0)
96	No Patch	1313.83	216.50	94656.61	0.00138	0.00525	0.0054
	No Patch (b-c)	208.63	151.09	2202.29	0.00161	0.00582	0.0063
	SQR9	6.68	2.32	150.86	0.00116	0.00544	0.0058
	SQR9 (b-c)	6.76	2.45	139.75	0.00091	0.00496	0.0049
	HOR9	1.90	1.52	16.75	0.00094	0.00528	0.0056
	HOR9 (b-c)	1.79	1.45	13.78	0.00117	0.00524	0.0056
	VER9	1.74	1.05	29.85	0.00129	0.00495	0.0048
	VER9 (b-c)	1.90	0.98	31.2	0.00137	0.00517	0.0051
	FRAC4	42.03	12.52	1463.5	0.00125	0.00576	0.0063
	FRAC4 (b-c)	28.98	10.96	848.91	0.00127	0.00558	0.00595
128	No Patch	3915.84	60.68	189627.72	0.00198	0.00442	0.0041
	No Patch (b-c)	1022.12	239.45	137694.3	0.00203	0.00547	0.0058
	SQR9	55.66	7.54	41190.94	0.00137	0.00545	0.00575
	SQR9 (b-c)	23.32	7.07	1415.85	0.00132	0.00522	0.0053
	HOR9	5.57	3.82	67.96	0.00126	0.00559	0.0061
	HOR9 (b-c)	4.59	3.4	41.77	0.00144	0.00565	0.0062
	VER9	5.56	3.26	85.06	0.00132	0.00518	0.0051
	VER9 (b-c)	5.79	3.13	61.13	0.00154	0.00539	0.0055

b-c. (VER9) again achieves the best run times with 28105.39 seconds with *b-c* and 31085.11 seconds without. We note that *b-c* performs better with regards to the run time for all decompositions finishing within the time limit which is in contrast to the case $N = 96$.

The different decomposition achieve different objective values which is in line with the convergence results as these do not guarantee a convergence to the same point. Furthermore, we see that even for the same decomposition different objectives are achieved. As the integer programs are only solved to the default Gurobi MIP gap, different optimal points can be returned which lead to different steps in the trust-region algorithm if accepted. In the previous numerical example using an advection-diffusion problem, we have not seen these effects in the results. We note that the run time results have to be taken with some caution due to this effect. The objective values between using *b-c* and not using *b-c* are very close for each decomposition which is sometimes in favor of *b-c* and sometimes not. The biggest gap for $N = 96$ occurs for (SQR9) in which using *b-c* produces a better objective value with 0.004194 and 0.004199 respectively. (SQR9) is the best performing decomposition with regards to the objective value while (VER9) performs the worst with objective values of 0.004220 with *b-c* and 0.004219 without *b-c*. The difference in objective values for $N = 96$ remains below one percent. The biggest difference in objective values for $N = 128$ is also below one percent and is between (SQR9) without *b-c* (objective value of 0.004196) and (HOR9) without *b-c* (objective value of 0.004233).

In Table 10.13 we have tabulated the mean, median and maximal run times for the integer programs occurring for each decomposition. We see that in all cases, except for $N = 96$ and (VER9) using b-c, the maximal run time is reduced by b-c. For the decomposition for which b-c reduces the run times we also see a decrease in the median run time. We note that (HOR9) has similiar mean and median run times compared to (VER9) but needs more outer iterations in the trust-region algorithm. Furthermore, we have tabled the mean and median gap which seem very close for every decomposition with regards to using b-c or not. We do not notice any particular outliers.

Interpretation of the numerical results

The numerical results again support the use of a decomposition approach to reduce the overall run time. The differences in objective values are barely noticable and do not impede the applicability of any decomposition. The use of the decomposition based on the fractional component (FRAC4) improves the run time compared to (No Patch) but the improvements are clearly worse than in the previous experiment. Together with the previous results, the usefulness of (FRAC4) instead of using 9 patches is not straightforward given the results. Furthermore, it currently needs the computation of a fractional component beforehand and thus is more an approach to validate the theoretical idea to use the findings regarding the fractional component for the decomposition rather than an advisable approach in practice without additional strategies. (For example, one could start with (No Patch) and then switch to (FRAC4) after encountering the first difficult to solve integer program while using the current iterate for initialization.)

We notice the desired effects for the decompositions into stripes. As in the previous experiments, the decompositions into stripes show a clear improvement in run times compared to a decomposition into squares. We argue that the decomposition into stripes should be very preferable to the decomposition into squares due to the run time gains despite the ever so slightly worse objective values.

The branching priority and the cutting planes show a consistent behaviour compared to our first experiments for the decompositions in Section 10.2.1. We see that they are able to improve the run times for approaches with long run times (and hard subproblems) but might increase the run time for decomposition which already reduce the run times significantly (as the subproblems also tend to be easier). In the case that $N = 128$ we also see all decompositions using 9 patches profiting from b-c compared to $N = 96$ for which none did. Additionally, they seem to be able to prevent very high run times of the individual integer program to a certain extend. The extension to the non-binary case looks promising from a superficial point of view but requires more indepth future research and more experiments similar to Section 10.1.2

on the same integer programs to be validated. This is beyond the scope of this work which focuses on the binary (TR-IP).

We can clearly see the anisotropic behaviour of the discretization in Figure 10.7. We have a lot of rectilinear shapes compared to the original picture. One can recognize the original structures but more pixels are needed for a better reconstruction. In contrast, the ideas of [91] produced pictures with structures closer to the original ones, see for example Figure 6.8 in [91]. The desired effect of multiple grey values is achieved such that the example was indeed useful for the evaluation of the cutting planes.

Figure 10.7: On the left the original picture of the dog Floki is displayed. In the middle the noisy image can be seen. On the right the final result for (SQR9) with (b-c) for $N = 128$ is displayed.

Chapter 11

Conclusion and outlook

The major focus of this dissertation was the two-dimensional (TR-IP) with a binary control set and an underlying rectangular domain which was uniformly discretized. We have shown how to reformulate the problem as an integer program and given several relaxations. We noted the connections to graph-based problems and conjectured the NP-hardness of the problem. We have gathered relevant insights regarding the polyhedron of the linear programming relaxation in Chapter 7. The results can be extended to more general cases. On the basis of these insights we have developed a variety of approaches to improve the run time of the integer programming solvers. Our numerical results support that the proposed approaches are able to significantly improve the run time. Especially the cutting planes proved to be a major benefit. The decompostion approach further solidified that it is advantageous to use approaches based on the fractional component.

We see three major avenues for potential future research. The first avenue is research with regards to extending the approaches to more general cases with a different underlying domain, a non-uniform discretization, a non-contiguous sets Ξ or a higher dimension. While some results and approaches will still be usable to some extent, we expect that there will be a need to adapt the approaches accordingly. The results regarding the fractional component should hold for all the extensions (dependent on the integer programming formulation). A different underlying domain would result in a different underlying graph structure. This would effect how the one-dimensional case (or the extension to trees) can be used to obtain good primal points. The bounding box cutting plane might not be applicable, however the cutting plane based on the fully connected graph should still work. A non-uniform discretization would introduce positive weights for the used capacity terms and for the jump terms in the objective. While the latter weights are not (as) problematic, the weights in the capacity constraints are. On the primal side we have already seen that these weights are integrated into the graph construction and the shortest path approach. The cutting planes would require significant overhaul to be applicable as the connection

to the minimum bisection problem is not straightforward anymore. In the case that Ξ is non-contiguous, we would need to give a more complicated integer programming reformulation, as the condition that the new control is also in Ξ is not fulfilled by the combination of the integrality condition and the upper/lower bounds on the optimization variables anymore. For a higher-dimensional (TR-IP) using the one-dimensional case to obtain good primal solutions is probably not warranted because too many jump terms would have to be dropped. For useful cutting planes one would need to obtain similar results with regards to the minimum bisection problem. We conjecture that using the idea of the fully connected graph as well as the bounding box approach and the resulting cutting planes should be transferable to some extend.

The second avenue of research is to even further improve the run time of the binary two-dimensional case with a uniform discretization of a rectangular domain. In the dissertation we have only used a branching priority. Using a proper branching rule might allow for a significant reduction in the size of the branching tree. We propose two ideas for possible branching rules based on the results of this dissertation.

Branching rule based on eigenvector centrality As we branch on integer variables with non-integer values in the linear programming relaxation, the branching candidates are given by the entries of the fractional component. We have already seen that the corresponding graph is a connected subgraph. We want to use a measurement for the connectivity of the graph and the importance of the individual node for our branching decision. In the literature we can find a variety of possible approaches based on the centrality of the nodes, see [70]. A very common approach is to use the eigenvector centrality, see for example [16], which we want to use in the following.

Let A be the adjacency matrix of the graph. An entry $a_{i,j} = 1$ if the nodes v_i and v_j are connected by an edge, otherwise $a_{i,j} = 0$. The centrality of a node is derived from the eigenvector equation $Av = \lambda v$ where λ is an eigenvalue and v a corresponding eigenvector. It follows from the well-known Perron-Frobenius theorem, see for example [83] for a small overview, that there exists an eigenvector to the largest eigenvalue which is non-negative in every entry. As the multiplication of an eigenvector with a scalar value retains the eigenvector property we choose a normalization of the eigenvector. The larger an entry of the eigenvector the more central the corresponding node is for the graph. In our branching rule we determine the fractional component and determine this eigenvector for the corresponding graph. We branch on the node with the largest entry. Ties are broken arbitrarily.

Branching rule based on the cut with the minimum cut ratio We have seen that we can calculate the minimum cut ratio and the corresponding cut efficiently for solid grid graphs. Not every fractional component induces a solid grid graph but we can add nodes to the graph corresponding to the fractional component until we obtain a solid grid graph. We calculate the cut with the minimum cut ratio. As

branching candidates we choose the nodes corresponding to entries in the fractional component which are also incident to a cut edge. By construction we can find at least one such node. Between the branching candidates the selected variable is chosen arbitrarily. (It might, however, prove beneficial to obtain an additional measurement for this selection process as well.) If all these nodes are fixed to integer values, the fractional component would be split into two sets of nearly equal sizes. As we have shown that the linear programming solution only has one fractional component, this would imply a significantly smaller fractional component. We note that with every branching decision the fractional component might change significantly. Furthermore, the new fractional component might contain both sets by also consisting of nodes adjacent to the original fractional component such that the branching rule does not result in the desired effect. Thus, the behaviour of the fractional component with regards to the branching rule has to be studied in detail which might give rise to even more sophisticated branching rules.

Furthermore, we have seen that the problem (F-TRIP) can be solved in polynomial time. This could open up avenues to construct cutting planes with arbitrary coefficients and also allow us to relax the assumption that the previous value has to be the same for the whole fully connected graph.

The final avenue of research is the combination of our measures with the discretization ideas from [92] to recover the correct total variation in the limit. As our problem is only the first integer program needed to solve in the solution approach, it is of particular interest if a similar property to the fractional component still exists and if some of the ideas for the cutting planes can be adapted to decrease the run time of later integer programs as well.

The picture of Floki used for the imaging problem and the picture of Nimue below, see Figure 11.1, may be used for research purposes.

Figure 11.1: This is Nimue.

Appendix A

Further explored avenues

In the dissertation we presented theoretical results and derived several ways to significantly reduce the computational demand of the trust-region algorithm and its subproblems. In this part of the appendix we will briefly present several additional avenues we explored which did not prove useful in preliminary computational experiments.

A.1 The k shortest path approach

For the constrained shortest path problem a k shortest path approach can be employed, see [54] and [36]. A feasible algorithm for the k shortest path problem is given in [38] and has a run time of $\mathcal{O}(m+n\log n+k)$. The constrained shortest path problem is solved by iteratively calculating the next best path with regards to the objective function. The algorithm in [38] only has a constant time demand for the calculation of the next path. If the calculated k shortest path fulfills the capacity constraint, it is straightforward to see the optimality for the constrained shortest path problem.

For the two-dimensional (TR-IP) one could employ an algorithm which calculates the next best cut until a cut is found which satisfies the capacity constraint. However, as far as we are aware there currently is no known algorithm which calculates the next best cut in constant time. An algorithm to calculate the k best cuts in $\mathcal{O}(kn^4)$ is given in [53]. Instead, we combine the presented dual decomposition from Section 4.3 with the k shortest path approach.

Let $\lambda \in \mathbb{R}^{N\times M}$ be fixed. For the sake of clarity, let $c(d)$ denote the objective value for a feasible d for the problem (IP) while $c_c(d)$ and $c_r(d)$ denote the objective values for (C-DD) and (R-DD) for the given λ respectively.

Theorem A.1. *Let $\lambda \in \mathbb{R}^{N\times M}$ be fixed. Let $R_k \coloneqq \{d^{r,1}, \dots, d^{r,k}\}$ denote the k best solutions to* (R-DD) *and $C_{\tilde{k}} \coloneqq \{d^{c,1}, \dots, d^{c,\tilde{k}}\}$ the best $\tilde{k}$ solutions to* (C-DD)*. If*

$$c_c(d^{r,k}) \leq c_c(d^{c,\tilde{k}}) \tag{A.1}$$

holds, then $R_k \cup C_{\tilde{k}}$ contains an optimal point for (IP)*.*

Proof. Assume that there does not exist a d in $R_k \cup C_{\tilde{k}}$ which is optimal for (IP). Let d^* be optimal for (IP) but not be in the set $R_k \cup C_{\tilde{k}}$ by assumption. It follows from (A.1) that $c_c(d^{r,k}) \leq c_c(d^*)$ holds. Furthermore, it follows that $c_r(d^{r,k}) \leq c_r(d^*)$ by the construction of R_k. Thus, it holds that

$$c(d^{r,k}) = c_c(d^{r,k}) + c_r(d^{r,k}) \leq c_c(d^*) + c_r(d^*) = c(d^*).$$

It follows that $d^{r,k}$ is optimal for (IP) which is a contradiction to the assumption. □

The theorem allows us to compute an optimal point for (IP) by iteratively calculating the shortest paths for (C-DD) and (R-DD) until the condition (A.1) is fulfilled. The algorithm is given in Algorithm 5. We start by calculating the shortest paths for both problems. Afterwards, we continue in a while loop. In the loop we calculate the next best path for (C-DD) or (R-DD) depending on the best previously found solution with regard to the objective function of (IP) (we choose the problem with the worse objective value). If the newly found path has a better objective value, we update the best previously found solution for (C-DD) or (R-DD) with regards to the objective function of (IP). We check if the condition (A.1) is fulfilled, which is the termination criteria derived from the previous theorem. In that case, we return the best found solution between (C-DD) or (R-DD). Otherwise, the while loop begins anew. The algorithm terminates in finite time as the number of feasible points for (IP), and thus for (C-DD) and (R-DD), are finite. The run time is linear in the number of calculated paths and can thus be exponential in the size of the grid.

A.2 Cutting planes based on the results in Section 5.4

In Section 5.4 we have seen that we can calculate the minimum cut ratio for solid grid graphs in polynomial time. In Section 8.3.3 we have constructed two cutting planes based on the minimum cut ratio for the fully connected graph and the bounding box. With the same underlying idea it is possible to construct a similar cutting plane if the fractional component is a solid grid graph (or we add the nodes needed such that the resulting graph is a solid grid graph). In preliminary experiments those cutting planes did not prove beneficial if the cutting planes detailed in Section 8.3.3 were already used. We do note that this still might present an avenue to create better cutting planes in the future or could prove beneficial if additional criteria are established for the selection process of the cutting planes.

Algorithm 5 dual decomposition approach combined with k shortest path approach

Input: Instance of (TR-IP), $\lambda \in \mathbb{R}^{N \times M}$

$k \leftarrow 1$
$\tilde{k} \leftarrow 1$
$d^c \leftarrow \tilde{k}$-th best solution to (C-DD)
$d^r \leftarrow k$-th best solution to (R-DD)
$k \leftarrow k + 1$
$\tilde{k} \leftarrow \tilde{k} + 1$
$b_c \leftarrow d^c$
$b_r \leftarrow d^r$
while $n = 1, \ldots$ **do**
    if $c(b_r) \leq c(b_c)$ **then**
        $d^c \leftarrow \tilde{k}$-th best solution (C-DD)
        $\tilde{k} \leftarrow \tilde{k} + 1$
        if $c(d^c) < c(b_c)$ **then**
            $b_c \leftarrow d^c$
        end if
        if $c_r(d^c) \leq c_r(d^r)$ **then**
            if $c(b_c) < c(b_r)$ **then**
                return b_c
            else
                return b_r
            end if
        end if
    else
        $d^r \leftarrow k$-th best solution (R-DD)
        $k \leftarrow k + 1$
        if $c(d^r) < c(b_r)$ **then**
            $b_r \leftarrow d^r$
        end if
        if $c_c(d^r) \leq c_c(d^c)$ **then**
            if $c(b_c) < c(b_r)$ **then**
                return b_c
            else
                return b_r
            end if
        end if
    end if
end while

A.3 Intersection cuts

In Section 8.3.3 we introduced cutting planes which allowed for a significant reduction in the run times of the subproblems. These cutting planes were based on the fractional

component property. The following cutting planes do not depend on this property but use the fact that the trust-region constraint uses an L^1 norm to construct a valid cutting plane.

The general idea of intersection cuts is to use a convex set S around the current linear programming solution which does not contain any (feasible) integer-valued points in the interior in order to construct a cutting plane through the intersection points of S and the polyhedron of the linear programming relaxation. We briefly present the general theoretic basis for the intersection cuts from [4]. For the sake of clarity, we assume non-degeneracy but the construction also works for the degenerate case and we refer the reader to [4] for the details. Afterwards, we discuss how to construct the set S based on the current linear programming relaxation solution.

We consider the feasible integer program

$$\text{(G-IP)} \qquad \begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b, \\ & x \geq 0, \\ & x \in \mathbb{Z}^n \end{aligned}$$

with $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. The linear programming relaxation in standard form reads

$$\text{(G-LP)} \qquad \begin{aligned} \min \quad & \tilde{c}^T t \\ \text{s.t.} \quad & \tilde{A} t = b, \\ & t \geq 0 \end{aligned}$$

with $t = (x, y)$, $\tilde{c} = (c^T, 0)^T$ and $\tilde{A} = (A, I)$.

Let $\bar{t} = (\bar{x}, \bar{y})$ be a non-degenerate vertex solution for (G-LP) which contains non-integer values in $\bar{x}$. Let I, J denote the index sets for the basic and non-basic variables respectively. Consider the simplex tableau

$$\begin{pmatrix} I & \bar{A} \end{pmatrix} \begin{pmatrix} t_I \\ t_J \end{pmatrix} = \begin{pmatrix} \bar{t}_I \\ \bar{t}_J \end{pmatrix}$$

for $\bar{t}$ with matrix $\bar{A}$ for the non-basic variables. It follows that

$$x_i = \bar{x}_i - \sum_{j \in J} \bar{a}_{i,j} t_j \quad \forall i \in I \cap \{1, \ldots, n\}$$

and

$$x_i = 0 \quad \forall i \in J \cap \{1, \ldots, n\}.$$

For each of the non-basic variables and the corresponding columns in the simplex tableau we obtain a halfline in the space of x with

$$x_i^j = \bar{x}_i - \bar{a}_{i,j}\lambda^j, \quad \lambda^j \geq 0, j \in J.$$

Due to the non-degeneracy assumption we know that each halfline contains an edge of the polyhedron. We define $r_i^j = -\bar{a}_{i,j}$ for $i \in I \cap \{1, \ldots, n\}$, $r_j^j = 1$ and $r_i^j = 0$ if $i \in J \cap \{1, \ldots, n\} \setminus \{j\}$. Let S be a convex set with $\bar{x} \in int(S)$ which does not contain a feasible point for (G-IP) in the interior. We determine

$$\lambda^{j,*} = \max \lambda^j \text{ s.t. } \bar{x} + r^j\lambda^j \in S.$$

We calculate a hyperplane through all the corresponding points $x^{j,*}$ with

$$x^{j,*} = \bar{x} + r^j\lambda^{j,*}.$$

The part of the polyhedron cut off by this hyperplane lies completely in S because S is convex. Thus, no feasible point for (G-IP) is cut off. A possible way to construct the cut is the formulation

$$\sum_{j \in J} \frac{1}{\lambda^{j,*}} t_j \geq 1,$$

with $\frac{1}{\infty} = 0$, see [29] Theorem 3.1. In our experiments we constructed the set S by rounding all d entries in our linear programming solution to the nearest integer. We denote the linear programming relaxation by d^{LP} and the rounding solution by d^R. Then

$$S = \{d \mid ||d - d^{LP}||_{L^1} \leq ||d^R - d^{LP}||_{L^1}\}$$

does not contain a feasible point of (G-IP) in the interior.

The resulting cuts did not improve the bounds by a significant margin and did not improve the run times. Improving the set S such that we round the linear programming relaxation solution to the nearest integer point, which also fulfills the trust-region constraint, did not prove effective either.

A.4 Branch-and-price for (IP)

A popular approach for integer programming is a branch-and-price algorithm which uses column generation to solve smaller linear programs, see [34]. The standard example for a case in which column generation works well is the cutting stock problem, see [47]. In this subsection we will use column generation in a branch-and-price algorithm on a Dantzig-Wolfe reformulation following the general theory detailed in [34]. We start by briefly presenting the needed theoretical results. Note that we will

only discuss the most essential parts and otherwise refer the reader to [34]. We start with an introduction of column generation and the branch-and-price algorithm based on the presentation in [34]. Afterwards, we discuss the Dantzig-Wolfe reformulation and how to incorporate this into the branch-and-price algorithm. We provide an explanation why our problem does not seem to be a good fit for this approach.

Column generation and branch-and-price Column Generation is an approach to solve linear programs in which only a subset of the columns, meaning a subset of the variables, are considered. Let the feasible linear program

$$\text{(MP)} \qquad \begin{aligned} \min_{\lambda} \quad & \sum_{j \in J} c_j \lambda_j \\ \text{s.t.} \quad & \sum_{j \in J} a_j \lambda_j \geq b, \\ & \lambda_j \geq 0 \quad \forall j \in J, \end{aligned}$$

which we refer to as the master problem be given with a (finite) index set J, $a_j, b \in \mathbb{R}^m$ and $c_j \in \mathbb{R}$ for all $j \in J$. The idea of column generation is to only use a subset $J' \subset J$ of the columns and solve the restricted master problem

$$\text{(RMP)} \qquad \begin{aligned} \min_{\lambda} \quad & \sum_{j \in J'} c_j \lambda_j \\ \text{s.t.} \quad & \sum_{j \in J'} a_j \lambda_j \geq b, \\ & \lambda_j \geq 0 \quad \forall j \in J', \end{aligned}$$

instead. The feasibility of the problem (RMP) depends on the choice of the subset J' but for now we assume that the problem is feasible and discuss how to actually ensure the feasibility later. Any point feasible for (RMP) directly gives us a feasible point for (MP) as we can set the missing entries to 0. Let (λ, π) be an optimal primal-dual pair for the problem (RMP). It is not guaranteed that the point we obtain by taking λ and setting the missing entries to 0 is optimal for (MP). From the theory of the simplex algorithm we know that the constructed point is optimal if the reduced costs $\bar{c}_j = c_j - \pi^T a_j$ are non-negative for all $j \in J$. If there exists a negative reduced cost entry, we add the column with the most negative value to J' and resolve the linear program. We repeat this process until all reduced cost entries are non-negative which gives us the optimality for (MP). We only need to add a finite amount of columns as J itself is finite. The problem of determining the column with the most negative reduced costs is called the pricing problem.

The branch-and-price algorithm is a variation of the branch-and-bound algorithm in which we use column generation to solve the occurring linear problems. We note that the branching process means that the initial set of columns in a node might

produce an infeasible linear program. In this case, the so-called Farkas pricing can be employed to restore feasibility. The Farkas pricing problem is identical to the presented pricing problem with the variation that we set $c_j = 0$ for all $j \in J$. For details we refer the reader to pages 51-53 in [34].

Dantzig-Wolfe reformulation We now turn to the Dantzig-Wolfe reformulation, see [33]. Let the problem

$$\text{(P)} \qquad \begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \\ & Dx \geq d \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

with $A \in \mathbb{R}^{m \times n}$, $D \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^m$, $d \in \mathbb{R}^k$ and $c \in \mathbb{R}^n$ be given. If a formulation of the integer hull $\text{conv}(\{x \in \mathbb{R}^n_{\geq 0} : Ax \geq b, Dx \geq d\} \cap \mathbb{Z}^n)$ as a polyhedron exists and is given, one can solve the problem (P) as a linear program on this polyhedron instead. We have mentioned this in Section 2.2 as well as the difficulties of this approach.

The idea for the reformulation is to use the properties of polyhedra described by the Minkowski-Weyl theorem, which states that a polyhedron can be characterized by its vertices and extreme rays instead of linear constraints, see [79] p. 155-156.

Theorem A.2 (Minkowski-Weyl, 7.3(h) in [79])**.** *A set $P \subset \mathbb{R}^n$ is a polyhedron if and only if there exist two finite set of points $S \subset \mathbb{R}^n$ and $T \subset \mathbb{R}^n$ such that $P = \text{conv}(S) + \text{cone}(T)$.*

A polytope can thus be characterized by its vertices, see Theorem 3.31 in [63]. This gives rise to the following theorem stated in Theorem 4.1 in [34].

Theorem A.3 (Theorem 4.1 in [34])**.** *Let $\mathcal{P} = \{x \in \mathbb{R}^n \mid Qx \geq q\} \neq \emptyset$ be a polyhedron with full row rank matrix $Q \in \mathbb{R}^{m \times n}$ with the integer set $\mathcal{Q} = \mathcal{P} \cap \mathbb{Z}^n \neq \emptyset$. Then*

$$\text{(A.2)} \qquad \mathcal{Q} = \left\{ x \in \mathbb{Z}^n \;\middle|\; \begin{aligned} \sum_{p \in P} x_p \lambda_p + \sum_{r \in R} x_r \lambda_r &= x, \\ \sum_{p \in P} \lambda_p &= 1, \\ \lambda_p &\geq 0 \quad \forall p \in P, \\ \lambda_r &\geq 0 \quad \forall r \in R \end{aligned} \right\},$$

where $\{x_p\}_{p \in P}$ are the extreme points and $\{x_r\}_{r \in R}$ are the extreme rays of $\text{conv}(\mathcal{Q})$.

Proof. See the proof of Theorem 4.1 in [34] in combination with the proof of Theorem 3.1 in [34] as well as Theorem 4.8 in [77]. □

In the case that $\text{conv}(\{x \in \mathbb{Z}^n \mid Dx \geq d\})$ is a polytope, the problem (P) can be reformulated using Theorem A.3 as

$$\begin{aligned} \min_{\lambda,x} \quad & \sum_{p\in P} c_p \lambda_p \\ \text{s.t.} \quad & \sum_{p\in P} a_p \lambda_p \geq b, \\ & \sum_{p\in P} \lambda_p = 1, \\ & \lambda_p \geq 0 \quad \forall p \in P, \\ & \sum_{p\in P} x_p \lambda_p = x \in \mathbb{Z}^n_{\geq 0}, \end{aligned} \tag{A.3}$$

where P is the set of extreme points of $\text{conv}(\{x \in \mathbb{Z}^n \mid Dx \geq d\})$ and $c_p = c^T x_p, a_p = Ax_p$, see [34] Chapter 4.

The problem (A.3) is solved with a branch-and price algorithm. The pricing problem is given as

$$\begin{aligned} \min_{x} \quad & (c^T - \pi_b^T A)x - \pi_0 \\ \text{s.t.} \quad & Dx \geq d, \\ & x \geq 0, \\ & x \in \mathbb{Z}^n \end{aligned} \tag{A.4}$$

where π_b is the dual vector to the inequalities $Ax \geq b$ and π_0 is the dual variable to the constraint $\sum_{p\in P} \lambda_p = 1$.

Branch-and-price for (IP) We now transfer this idea to the binary version of (IP) in the reformulated form

$$\begin{aligned} \min_{\delta,\beta,\gamma} \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}(-1)^{x_{i,j}}\delta_{i,j} + \alpha\left(\sum_{j=1}^{M}\sum_{i=1}^{N-1}\beta_{i,j} + \sum_{i=1}^{N}\sum_{j=1}^{M-1}\gamma_{i,j}\right) \\ \text{s.t.} \quad & (\delta,\beta,\gamma) \in \tilde{P}_\Delta \text{ and } \delta \in \mathbb{N}^{N\times M}, \end{aligned} \tag{A.5}$$

where $\tilde{P}_\Delta = \{(\delta, \beta, \gamma) \in \tilde{P} : \sum_{i=1}^{N} \sum_{j=1}^{M} \delta_{i,j} \leq \Delta\}$ is the polyhedron obtained from the intersection of the capacity constraint and $\tilde{P}$ defined by

$$(\delta, \beta, \gamma) \in \tilde{P} \quad \Leftrightarrow \begin{cases} 0 \leq u_{i,j} + (-1)^{u_{i,j}} \delta_{i,j} \leq 1 \\ \text{ for all } i \in [N], j \in [M], \\ -\beta_{i,j} \leq u_{i+1,j} + (-1)^{u_{i+1,j}} \delta_{i+1,j} - u_{i,j} - (-1)^{u_{i,j}} \delta_{i,j} \leq \beta_{i,j} \\ \text{ for all } i \in [N-1], j \in [M], \\ -\gamma_{i,j} \leq u_{i,j+1} + (-1)^{u_{i,j+1}} \delta_{i,j+1} - u_{i,j} - (-1)^{u_{i,j}} \delta_{i,j} \leq \gamma_{i,j} \\ \text{ for all } i \in [N], j \in [M-1], \\ \beta_{i,j} \leq 1 \text{ for all } i \in [N-1], j \in [M], \\ \gamma_{i,j} \leq 1 \text{ for all } i \in [N], j \in [M-1], \end{cases}$$

such that we need only non-negative and fewer variables compared to the original formulation of (IP) as

$$d_{i,j} = (-1)^{u_{i,j}} \delta_{i,j} = \begin{cases} \delta_{i,j}, & u_{i,j} = 0, \\ -\delta_{i,j}, & u_{i,j} = 1. \end{cases}$$

Note that we have introduced the constraints $\beta_{i,j} \leq 1$ and $\gamma_{i,j} \leq 1$ to ensure that $\tilde{P}$ is a polytope as all optimization variables are now bounded from below by 0 and from above by 1. This does not change the optimization problem as an optimal point can not contain values larger than 1. We use the inequalities describing $\tilde{P}$ as the equivalent to the set of inequalities $Dx \geq d$ and the capacity constraint as the equivalent to $Ax \geq b$. Since $\tilde{P}$ is a polytope, we obtain that $\text{conv}(\tilde{P} \cap \mathbb{Z}^{N \times M + (N-1) \times M + N \times (M-1)})$ is also a polytope as this integer hull is a convex hull of finitely many points, see Theorem 3.31 and Chapter 5 in [63]. The pricing problem is the binary version of the inner problem of the Lagrangian relaxation (LR-Δ) with $\mu = \pi_b$ and $\mu\Delta$ replaced by π_0.

$$\begin{aligned} \min_{\delta, \beta, \gamma} \quad & \sum_{i=1}^{N} \sum_{j=1}^{M} c_{i,j} (-1)^{x_{i,j}} \delta_{i,j} + \alpha \left(\sum_{j=1}^{M} \sum_{i=1}^{N-1} \beta_{i,j} + \sum_{i=1}^{N} \sum_{j=1}^{M-1} \gamma_{i,j} \right) + \pi_b \sum_{i=1}^{N} \sum_{j=1}^{M} \delta_{i,j} - \pi_0 \\ \text{s.t.} \quad & (\delta, \beta, \gamma) \in \tilde{P} \text{ and } \delta \in \mathbb{Z}^{N \times M}. \end{aligned}$$

As shown, this problem can be solved efficiently as a minimum s-t cut problem.

Implementation We have implemented the formulation (A.3) in the branch-and-price framework of SCIP 8.1.0 [14] in Python using PySCIPOpt [71]. We branch on the original variables δ. As detailed above the pricing problems are minimum s-t cut problems which we solve in C++ with the boost graph implementation [17] of the

s-t minimum cut approach from [18] which we access with Pybind11 [60]. We note that we have implemented the model with the constraints $\sum_{p \in P} \delta_p \lambda_p = \delta \in \mathbb{Z}^n_{\geq 0}$ like the last set of constraints of (A.3) so we can branch on these variables.

We initialize the set of columns with the two integer points, which we obtain from rounding the solution to the linear relaxation. We run the instances for $N = M \in \{32, 64\}$ and $\alpha = 4\sqrt{5} \times 10^{-4}$ from [74] and compare the run times of the branch-and-price approach to the run times of solving the instances with SCIP for the original formulation. The underlying linear programming solver is SoPlex. We set a time limit of 30 minutes. We have also initially implemented the model in GCG [43] and used the automatic detection feature for the reformulation. This did not produce run times comparable to the original formulation, as they were significantly larger even for small instances, so we did not further explore this avenue. In order to ensure that the run times are not negatively effected by a suboptimal Python implementation we subtract the time in the callback functions except the time needed for the s-t cut calculation. We also supply the two solutions to the Lagrangian relaxation in the root node and do not consider the time to calculate these. This approach obviously favors the branch-and-price approach.

Table A.1: Given are the total times in seconds for the branch-and-price approach and directly using SCIP. We have further given the times for the s-t cut generation and the time spent outside of the callback functions for the branch-and-price approach.

N	branch-and-price	s-t cuts	time outside callback	SCIP
32	1738.98	972.11	766.87	95.77
64	29443.71	24166.12	5277.59	3675.99

Results The run times are detailed in Table A.1. We see that the total run time for the branch-and-price approach is several magnitudes higher than using SCIP on the original formulation. The time for the s-t cut calculation and the time spend outside of the callback functions are both larger than the run time of SCIP for the original formulation. Furthermore, for $N = 64$ the branch-and-price did not solve 20 out of 38 instances with an average gap of 0.093.

These preliminary numerical results hint that this approach might not be ideal as several implementation parts need to be significantly improved to be even comparable to just using the integer programming solver directly. We need a more problem specific implementation to solve the minimum s-t cut problem to significantly reduce the run time. This alone would, however, not suffice to warrant this approach as we would also need to reduce the time spend outside the callback function. One would need to implement the model in such a way that the constraints $\sum_{p \in P} \delta_p \lambda_p = \delta \in \mathbb{Z}^n_{\geq 0}$ are only added when we branch on the specific variable if possible. The approach

also has to include branching strategies as the current approach does not branch efficiently. From a theoretical standpoint we give some arguments against this branch-and-price approach for the problem (IP) in the following.

Arguments against this approach The numerical results do not support a further pursuit of this theoretical avenue. We also want to discuss from a theoretical standpoint why the problem does not seem to be a good case for this approach. In general it holds that the objective value of the linear relaxation of (IP) is a lower bound for the objective value for the linear relaxation of the master problem obtained from the reformulation, see Proposition 4.1 in [34] (this holds in every node in the branching tree as long as we branch on the original variables). In our case however, these are always identical. Let $(d, \delta, \beta, \gamma)$ be the solution to the linear relaxation of (IP). Let $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma})$ and $(\underline{d}, \underline{\delta}, \underline{\beta}, \underline{\gamma})$ be the integer points we obtain by rounding $(d, \delta, \beta, \gamma)$ up or down respectively. As the solution to the linear relaxation of (IP) only has one fractional component with the same value, we can obtain $(d, \delta, \beta, \gamma)$ as a convex combination of $(\bar{d}, \bar{\delta}, \bar{\beta}, \bar{\gamma})$ and $(\underline{d}, \underline{\delta}, \underline{\beta}, \underline{\gamma})$. Thus, the objective values of both linear relaxations coincide as the equivalent integer points in the master problem formulation can be added by column generation. In general one would hope to obtain a polyhedron closer to the integer hull compared to the polyhedron for the linear programming relaxation to the original integer program. However, from the results in Chapter 7, specifically Corollary 7.6, we know that all vertices of $\tilde{P}$ are already integer-valued and thus $\tilde{P}$ is identical to its integer hull as we explain in the following.

Let $(\tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ be a vertex of $\tilde{P}$. From Corollary 7.6 we know that $\tilde{\delta} \in \{0,1\}^{N \times M}$. Assume that we can find a $\tilde{\beta}_{i,j} \in (0,1)$ for some i, j. Then $u_{i+1,j} + (-1)^{u_{i+1,j}} \tilde{\delta}_{i+1,j} - u_{i,j} - (-1)^{u_{i,j}} \tilde{\delta}_{i,j} = 0$. Thus, we can construct two feasible points for $\tilde{P}$ by rounding the entry $\tilde{\beta}_{i,j}$ to 0 and 1 respectively while keeping all other entries the same. This would mean that $(\tilde{\delta}, \tilde{\beta}, \tilde{\gamma})$ is a convex combination of two feasible points and not a vertex. In turn, all entries have to be integer-valued (to be more precise binary).

It follows that we would always need to include the trust-region constraint into the constraint set $Dx \geq d$ in the construction to be able to expect a benefit. To ensure that the pricing problem is easy to solve, we need a construction where the pricing problem can only include jump terms such that the underlying graph is a path or tree.

Even if we were to only obtain such pricing problems, we might not get significant improvements if the remaining jumps are completely ignored. We have already talked in Example 4.14 that we often obtain $(d, \delta, \beta, \gamma)$ as a convex combination of feasible integer-valued points for such structures. If $(d, \delta, \beta, \gamma)$ can be obtained as a convex combination, we do not get a better bound from our branch-and-price approach. If $(d, \delta, \beta, \gamma)$ can not be obtained as a convex combination, then we may just obtain the same benefits for our original formulation by adding the lower bound for the

substructure as a cutting plane to our integer programming formulation as discussed in Section 4.3. So a construction also has to consider this aspect. Due to these reasons we decided against a further pursuit of a branch-and-price approach of this form and instead focused on the avenues presented in the dissertation but note that an approach which fulfills the above requirements might prove beneficial.

Bibliography

[1] Martin S. Alnaes, Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40, 2014. (cited on page 124.)

[2] Luigi Ambrosio, Nicola Fusco, and Diego Pallara. *Functions of Bounded Variation and Free Discontinuity Problems*. Oxford Mathematical Monographs. Clarendon Press Oxford, 2000. (cited on pages 18, 19, and 24.)

[3] Amitai Armon and Uri Zwick. Multicriteria global minimum cuts. *Algorithmica*, 46:15–26, 2006. (cited on page 51.)

[4] Egon Balas. Intersection cuts-a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39, 1971. (cited on page 148.)

[5] Egon Balas. *Disjunctive Programming*. Springer Cham, 1st edition, 2018. (cited on page 18.)

[6] Robert Baraldi and Paul Manns. Domain decomposition for integer optimal control with total variation regularization. *arXiv preprint arXiv:2410.15672, v3, accepted in SIAM Journal on Control and Optimization*, 2024. (cited on pages 12, 23, 24, 107, 108, 109, 110, 111, 129, 130, 131, and 134.)

[7] Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. DOLFINx: the next generation FEniCS problem solving environment. preprint, 2023. (cited on pages 124 and 130.)

[8] Sören Bartels. Total variation minimization with finite elements: Convergence and iterative solution. *SIAM Journal on Numerical Analysis*, 50(3):1162–1180, 2012. (cited on page 27.)

[9] B. Berkels, M. Burger, M. Droske, O. Nemitz, and M. Rumpf. Cartoon extraction based on anisotropic image classification. In *Vision, Modeling, and Visualization Proceedings*, pages 293–300, 2006. (cited on page 27.)

[10] D. P. Bertesakes. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999. (cited on page 29.)

[11] Felix Bestehorn, Christoph Hansknecht, Christian Kirches, and Paul Manns. A switching cost aware rounding method for relaxations of mixed-integer optimal control problems. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 7134–7139. IEEE, 2019. (cited on page 9.)

[12] Felix Bestehorn, Christoph Hansknecht, Christian Kirches, and Paul Manns. Mixed-integer optimal control problems with switching costs: a shortest path approach. *Mathematical Programming*, 188(2):621–652, 2021. (cited on pages 9, 122, and 123.)

[13] Felix Bestehorn, Christoph Hansknecht, Christian Kirches, and Paul Manns. Switching cost aware rounding for relaxations of mixed-integer optimal control problems: the two-dimensional case. *IEEE Control Systems Letters*, 6:548–553, 2021. (cited on page 9.)

[14] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021. (cited on page 153.)

[15] Lorena Bociu, Paul Manns, Marvin Severitt, and Sarah Strikwerda. Input regularization for integer optimal control in bv with applications to control of poroelastic and poroviscoelastic systems. *Journal of Nonsmooth Analysis and Optimization*, 5, 4 2024. (cited on page 13.)

[16] Phillip Bonacich. Some unique properties of eigenvector centrality. *Social Networks*, 29(4):555–564, 2007. (cited on page 142.)

[17] Boost. Boost C++ Libraries. `http://www.boost.org/`. Version 1.71.0. (cited on page 153.)

[18] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. (cited on pages 45 and 154.)

[19] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, pages 648–655, 1998. (cited on page 10.)

[20] Kristian Bredies and Dirk Lorenz. *Mathematical Image Processing*. Birkhäuser Cham, 1st edition, 2019. (cited on page 18.)

[21] Christoph Buchheim. Extended formulations for binary optimal control problems. *Mathematical Programming*, 11 2024. (cited on page 44.)

[22] Christoph Buchheim, Alexandra Grütering, and Christian Meyer. Parabolic optimal control problems with combinatorial switching constraints, part I: Convex relaxations. *SIAM Journal on Optimization*, 34(2):1187–1205, 2024. (cited on page 9.)

[23] Christoph Buchheim, Alexandra Grütering, and Christian Meyer. Parabolic optimal control problems with combinatorial switching constraints, part II: Outer approximation algorithm. *SIAM Journal on Optimization*, 34(2):1295–1315, 2024. (cited on page 9.)

[24] Christoph Buchheim, Alexandra Grütering, and Christian Meyer. Parabolic optimal control problems with combinatorial switching constraints, part III: branch-and-bound algorithm. *Computational Optimization and Applications*, 90(3):649–689, 2025. (cited on page 9.)

[25] Martin Burger, Yiqiu Dong, and Michael Hintermüller. Exact relaxation for classes of minimization problems with binary constraints. *arXiv preprint arXiv:1210.7507, v1*, 2012. (cited on page 77.)

[26] Valentina Cacchiani, Manuel Iori, Alberto Locatelli, and Silvano Martello. Knapsack problems — an overview of recent advances. part II: Multiple, multidimensional, and quadratic knapsack problems. *Computers and Operations Research*, 143, 2022. (cited on page 52.)

[27] E. Casas, K. Kunisch, and C. Pola. Regularization by functions of bounded variation and applications to image enhancement. *Applied Mathematics and Optimization*, 40:229–257, 1999. (cited on page 27.)

[28] Antonin Chambolle. Total variation minimization and a class of binary MRF models. In Anand Rangarajan, Baba Vemuri, and Alan L. Yuille, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 136–152, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. (cited on page 10.)

[29] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Corner polyhedron and intersection cuts. *Surveys in Operations Research and Management Science*, 16(2):105–120, 2011. (cited on page 149.)

[30] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009. (cited on pages 79, 80, 84, and 113.)

[31] Giacomo Cristinelli, José A. Iglesias, and Daniel Walter. Conditional gradients for total variation regularization with pde constraints: a graph cuts approach. *arXiv preprint arXiv: 2310.19777, v2*, 2024. (cited on page 27.)

[32] George B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–277, 1957. (cited on page 51.)

[33] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. (cited on page 151.)

[34] Jacques Desrosiers, Marco Lübbecke, Guy Desaulniers, and Jean Bertrand Gauthier. Branch-and-price. Les Cahiers du GERAD G-2024-36, Groupe d'études et de recherche en analyse des décisions, GERAD, Montréal QC H3T 2A7, Canada, October 2024. (cited on pages 149, 150, 151, 152, and 155.)

[35] Reinhard Diestel. *Graph Theory*. Springer Berlin, Heidelberg, 6th edition, 2025. (cited on pages 15, 16, and 17.)

[36] Irina Dumitrescu and Natashia Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks: An International Journal*, 42(3):135–153, 2003. (cited on pages 81, 82, and 145.)

[37] Josep Díaz and George B. Mertzios. Minimum bisection is NP-hard on unit disk graphs. *Information and Computation*, 256:83–92, 2017. (cited on page 64.)

[38] David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998. (cited on page 145.)

[39] Selim Esedoḡlu and Stanley Osher. Decomposition of images by the anisotropic Rudin-Osher-Fatemi model. *Communications on Pure and Applied Mathematics*, 57:1609 – 1626, 12 2004. (cited on page 27.)

[40] Lawrence C. Evans and Ronald F. Gariepy. *Measure theory and fine properties of functions*. Chapman and Hall/CRC, New York, revised edition(1st ed.) edition, 2015. (cited on page 25.)

[41] Andreas Emil Feldmann and Peter Widmayer. An $\mathcal{O}(n^4)$ time algorithm to compute the bisection width of solid grid graphs. *Algorithmica*, 71:181–200, 2015. (cited on pages 53, 61, and 87.)

[42] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020. (cited on pages 113 and 115.)

[43] Gerald Gamrath and Marco E. Lübbecke. Experiments with a generic dantzig-wolfe decomposition for integer programs. In Paola Festa, editor, *Symposium on Experimental Algorithms (SEA 2010)*, volume 6049 of *Lecture Notes in Computer Science*, pages 239–252, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. (cited on page 154.)

[44] Micheal R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979. (cited on page 40.)

[45] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984. (cited on page 10.)

[46] A. M. Geoffrion. Lagrangean relaxation for integer programming. In *Mathematical Programming Study 2*, pages 82–114. Springer, 1974. (cited on pages 34 and 37.)

[47] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961. (cited on page 149.)

[48] Simone Göttlich, Oliver Kolb, and Sebastian Kühn. Optimization for a special class of traffic flow models: Combinatorial and continuous approaches. *Networks & Heterogeneous Media*, 9(2):315–334, 2014. (cited on page 9.)

[49] Simone Göttlich, Andreas Potschka, and Ute Ziegler. Partial outer convexification for traffic light optimization in road networks. *SIAM Journal on Scientific Computing*, 39(1):B53–B75, 2017. (cited on page 9.)

[50] Jack E. Graver. On the foundations of linear and integer linear programming I. *Mathematical Programming*, 9(1):207–226, 1975. (cited on page 10.)

[51] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. (cited on pages 124 and 131.)

[52] Oliver Habeck, Marc E. Pfetsch, and Stefan Ulbrich. Global optimization of mixed-integer ODE constrained network problems using the example of stationary gas transport. *SIAM Journal on Optimization*, 29(4):2949–2985, 2019. (cited on page 9.)

[53] Horst Hamacher. An $\mathcal{O}(k{\cdot}n^4)$ algorithm for finding the k best cuts in a network. *Operations Research Letters*, 1(5):186–189, 1982. (cited on page 145.)

[54] Gabriel Y. Handler and Israel Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10(4):293–309, 1980. (cited on pages 40 and 145.)

[55] Falk M Hante, Günter Leugering, Alexander Martin, Lars Schewe, and Martin Schmidt. Challenges in optimal control problems for gas and fluid flow in networks of pipes and canals: From modeling to industrial applications. In *Industrial Mathematics and Complex Systems*, pages 77–122. Springer, 2017. (cited on page 9.)

[56] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. (cited on page 80.)

[57] Jaroslav Haslinger and Raino AE Mäkinen. On a topology optimization problem governed by two-dimensional Helmholtz equation. *Computational Optimization and Applications*, 62(2):517–544, 2015. (cited on page 9.)

[58] Michael Hinze, René Pinnau, Michael Ulbrich, and Stefan Ulbrich. *Optimization with PDE constraints*, volume 23. Springer Science & Business Media, 2008. (cited on page 114.)

[59] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982. (cited on page 17.)

[60] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between C++11 and Python, 2017. `https://github.com/pybind/pybind11`. (cited on pages 124, 131, and 154.)

[61] Christian Kirches, Paul Manns, and Stefan Ulbrich. Compactness and convergence rates in the combinatorial integral approximation decomposition. *Mathematical Programming*, 188(2):569–598, 2021. (cited on pages 9 and 120.)

[62] Claus Kirchner, Michael Herty, Simone Göttlich, and Axel Klar. Optimal control for continuous supply network models. *Networks and Heterogeneous Media*, 1(4):675–688, 2006. (cited on page 9.)

[63] Bernhard Korte and Jens Vygen. *Combinatorial Optimization.* Springer, 6th edition, 2018. (cited on pages 15, 16, 17, 39, 45, 51, 76, 80, 151, and 153.)

[64] Drew P. Kouri. *An Approach for the Adaptive Solution of Optimization Problems Governed by Partial Differential Equations with Uncertain Coefficients.* PhD thesis, Rice University, 2012. (cited on page 114.)

[65] Michał Łasica, Salvador Moll, and Piotr B. Mucha. Total variation denoising in ℓ^1 anisotropy. *SIAM Journal on Imaging Sciences*, 10(4):1691–1723, 2017. (cited on page 27.)

[66] Sven Leyffer and Paul Manns. Sequential linear integer programming for integer optimal control with total variation regularization. *ESAIM: COCV*, 28:66, 2022. (cited on pages 10, 21, 22, and 23.)

[67] Sven Leyffer, Paul Manns, and Malte Winckler. Convergence of sum-up rounding schemes for cloaking problems governed by the Helmholtz equation. *Computational Optimization and Applications*, 79(1):193–221, 2021. (cited on pages 9 and 113.)

[68] Yuan Liang and Gengdong Cheng. Topology optimization via sequential integer programming and canonical relaxation algorithm. *Computer Methods in Applied Mechanics and Engineering*, 348:64–96, 2019. (cited on page 9.)

[69] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989. (cited on page 115.)

[70] Linyuan Lü, Duanbing Chen, Xiao-Long Ren, Qian-Ming Zhang, Yi-Cheng Zhang, and Tao Zhou. Vital nodes identification in complex networks. *Physics Reports*, 650:1–63, 2016. Vital nodes identification in complex networks. (cited on page 142.)

[71] Stephen Maher, Matthias Miltenberger, João Pedro Pedroso, Daniel Rehfeldt, Robert Schwarz, and Felipe Serrano. PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pages 301–307. Springer International Publishing, 2016. (cited on page 153.)

[72] Paul Manns, Mirko Hahn, Christian Kirches, Sven Leyffer, and Sebastian Sager. On convergence of binary trust-region steepest descent. *Journal of Nonsmooth Analysis and Optimization*, 2022. (cited on page 23.)

[73] Paul Manns and Annika Schiemann. On integer optimal control with total variation regularization on multidimensional domains. *SIAM Journal on Con-*

trol and Optimization, 61(6):3415–3441, 2023. (cited on pages 10, 19, 21, 23, 24, and 111.)

[74] Paul Manns and Marvin Severitt. On discrete subproblems in integer optimal control with total variation regularization in two dimensions. *INFORMS Journal on Computing*, 37(4):1121–1141, 2024. (cited on pages 10, 13, 29, 30, 32, 33, 35, 36, 37, 44, 63, 64, 66, 68, 69, 71, 72, 75, 76, 77, 79, 90, 92, 99, 101, 102, 111, 113, 125, 126, 127, 128, 129, and 154.)

[75] Jonas Marko and Gerd Wachsmuth. Integer optimal control problems with total variation regularization: Optimality conditions and fast solution of subproblems. *ESAIM: Control, Optimisation and Calculus of Variations*, 29:81, 2023. (cited on page 79.)

[76] R. R. Meyer. On the existence of optimal solutions to integer and mixed-integer programming problems. *Math. Program.*, 7(1):223–235, December 1974. (cited on page 17.)

[77] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, USA, 1988. (cited on page 152.)

[78] Claudia Nieuwenhuis, Eno Töppe, and Daniel Cremers. A survey and comparison of discrete and continuous multi-label optimization approaches for the potts model. *International Journal of Computer Vision*, 104(3):223–240, 2013. (cited on page 10.)

[79] Manfred Padberg. *Linear Optimization and Extensions*. Springer Berlin, Heidelberg, 2nd edition, 1999. (cited on page 151.)

[80] Christos H. Papadimitriou and Martha Sideri. The bisection width of grid graphs. *Mathematical Systems Theory*, 29:97–110, 1996. (cited on pages 17, 53, 55, 56, 57, and 64.)

[81] Christos H. Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 86–92. IEEE, 2000. (cited on page 51.)

[82] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., USA, 1984. (cited on page 80.)

[83] S.U. Pillai, T. Suel, and Seunghun Cha. The Perron-Frobenius theorem: some of its applications. *IEEE Signal Processing Magazine*, 22(2):62–75, 2005. (cited on page 142.)

[84] R. B. Potts. Some generalized order-disorder transformations. *Mathematical Proceedings of the Cambridge Philosophical Society*, 48(1):106–109, 1952. (cited on page 10.)

[85] Florian Rathgeber, David A. Ham, Lawrence Mitchell, Michael Lange, Fabio Luporini, Andrew T. T. McRae, Gheorghe-Teodor Bercea, Graham R. Markall, and Paul H. J. Kelly. Firedrake: automating the finite element method by composing abstractions. *ACM Trans. Math. Softw.*, 43(3):24:1–24:27, 2016. (cited on page 114.)

[86] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992. (cited on pages 10 and 27.)

[87] Stuart Russell and Peter Norvig. *Artificial Intelligence, Global Edition A Modern Approach*. Pearson Deutschland, 4th edition, 2021. (cited on page 80.)

[88] Sebastian Sager. *Numerical methods for mixed-integer optimal control problems*. Der Andere Verlag, Lübeck, 2005. (cited on page 121.)

[89] Sebastian Sager, Michael Jung, and Christian Kirches. Combinatorial integral approximation. *Mathematical Methods of Operations Research*, 73(3):363–380, 2011. (cited on pages 9 and 120.)

[90] Sebastian Sager and Clemens Zeile. On mixed-integer optimal control with constrained total variation of the integer control. *Computational Optimization and Applications*, 78(2):575–623, 2021. (cited on page 9.)

[91] Annika Schiemann. *Integer optimization with total variation regularization*. PhD thesis, TU Dortmund University (Universitätsbibliothek Dortmund), Dortmund, 2024. (cited on pages 135, 136, and 140.)

[92] Annika Schiemann and Paul Manns. Discretization of total variation in optimization with integrality constraints. *SIAM Journal on Numerical Analysis*, 63(1):437–460, 2025. (cited on pages 27, 28, 136, and 143.)

[93] Matthew W. Scroggs, Igor A. Baratta, Chris N. Richardson, and Garth N. Wells. Basix: A runtime finite element basis evaluation library. *Journal of Open Source Software*, 7(73):3982, 2022. (cited on page 124.)

[94] Matthew W. Scroggs, Jørgen S. Dokken, Chris N. Richardson, and Garth N. Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Transactions on Mathematical Software*, 48(2):18:1–18:23, 2022. (cited on page 124.)

[95] Marvin Severitt. Kombinatorische Optimierung für Trust-Region-Subprobleme im Kontext ganzzahliger Optimalsteuerungsprobleme, Master Thesis, 2021. (cited on pages 12, 39, 63, and 79.)

[96] Marvin Severitt and Paul Manns. Efficient solution of discrete subproblems arising in integer optimal control with total variation regularization. *INFORMS Journal on Computing*, 35(4):869–885, 2023. (cited on pages 10, 12, 30, 33, 39, 42, 43, 63, 64, 79, 82, 83, 84, 85, 111, 113, 114, 116, 117, 118, 119, 120, 121, 122, and 123.)

[97] Krister Svanberg and Mats Werme. Sequential integer programming methods for stress constrained topology optimization. *Structural and Multidisciplinary Optimization*, 34(4):277–299, 2007. (cited on page 9.)

[98] C. Umans and W. Lenhart. Hamiltonian cycles in solid grid graphs. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 496–505, 1997. (cited on page 17.)

[99] Olga Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, USA, 1999. (cited on page 45.)

[100] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. (cited on page 115.)

[101] Laurence A. Wolsey. Faces for a linear inequality in 0–1 variables. *Mathematical Programming*, 8(1):165–178, 1975. (cited on page 52.)

[102] Ying Xiao, K. Thulasiraman, Guoliang Xue, Alpár Jüttner, and S. Arumugam. The constrained shortest path problem: Algorithmic approaches and an algebraic study with generalization. *AKCE International Journal of Graphs and Combinatorics*, 2, 01 2005. (cited on page 82.)

[103] Dominic Yang. A specialized simplex algorithm for budget-constrained total variation-regularized problems. *arXiv preprint arXiv:2507.13493, v2*, 2025. (cited on page 10.)

[104] Dominic Yang, Sven Leyffer, and Miles Bakenhus. Augmentation algorithms for total variation-regularized integer programs. *arXiv preprint arXiv: 2508.05822, v2*, 2025. (cited on pages 10 and 33.)

[105] Clemens Zeile, Nicolò Robuschi, and Sebastian Sager. Mixed-integer optimal control under minimum dwell time constraints. *Math. Program.*, 188(2):653–694, August 2021. (cited on page 44.)