I N A U G U R A L – D I S S E R T A T I O N

zur

Erlangung der Doktorwürde

der

Naturwissenschaflich-Mathematischen Gesamtfakultät

der

Ruprecht-Karls-Universität

Heidelberg

vorgelegt von

Diplom Mathematiker	Dirk Oliver Theis
aus	Schwalmstadt

Tag der mündlichen Prüfung:	20.12.2005

Thema

Polyhedra and algorithms for the General Routing Problem

Gutachter:	Prof. Dr. Gerhard Reinelt
	Prof. Dr. Dr. h. c. Hans Georg Bock

Zusammenfassung

Das General Routing Problem ist ein auf ungerichteten Graphen definiertes **NP**-schweres kombinatorisches Optimierungsproblem. Es handelt sich um eine geringfügige Verallgemeinerung des besser bekannten Rural Postman Problem (siehe z.B. Garey & Johnson 1979, [GJ79]), zu dem es tatsächlich sowohl theoretisch als auch was die praktische Lösung betrifft äquivalent ist. Der Unterschied in der Namensgebung ist nur historisch bedingt.

Ein erfolgreicher Ansatz, um kombinatorische Optimierungsprobleme dieser Art praktisch zu lösen, d.h. unter der Menge der möglichen Lösungen die beste zu finden, ist der folgende. Die Menge der Lösungen einer konkreten Probleminstanz wird mit den Ecken eines konvexen Polyeders in einem euklidischen Vektorraum identifiziert. Ein konvexes Polyeder ist der Schnitt endliche vieler Halbräume. Ist eine Beschreibung des Polyeders durch lineare Ungleichungen gegeben, kann dann mit Standardmethoden der linearen Optimierung schnell die beste Lösung gefunden werden.

Um diesen Ansatz zu benutzen, müssen zwei Fragestellungen angegangen werden. Erstens müssen polyedrische Untersuchungen der Struktur des kombinatorischen Optimierungsproblems vorgenommen werden. Sowohl die Dimension des umgebenden euklidischen Raumes als auch die Struktur des Polyeders selber hängen nämlich ab von den Daten, welche die Probleminstanz definieren. In unserem Fall ist das der Graph. Ziel der Untersuchungen ist es, Klassen von Ungleichungen zu finden, die benötigt werden, um die Polyeder zu beschreiben. Diesen Zweig der diskreten Optimierung nennt man polyedrische Kombinatorik. Zweitens müssen die Klassen von linearen Ungleichungen, die man identifiziert hat, algorithmisch nutzbar gemacht werden. Das führt zur Entwicklung von Algorithmen, die das folgende Problem bearbeiten: Gegeben einen Punkt des umgebenden Raumes, stelle fest, ob er in dem Polyeder liegt, das durch die vorliegende Probleminstanz definiert wird, und falls das nicht der Fall ist, erzeuge eine Hyperebene in Form einer linearen Ungleichung, die den Punkt vom Polyeder trennt.

Diese Arbeit widmet sich sowohl der polyedrischen Kombinatorik als auch der algorithmischen Nutzbarmachung von Ungleichungen im Zusammenhang mit dem General Routing Problem. Zum ersten Punkt tragen wir strukturelle Eigenschaften der Polyeder sowie eine Reihe von bisher nicht bekannten Klassen von Ungleichungen bei. Für den zweiten Punkt stellen wir Algorithmen vor, die die oben beschriebenen Trennungsprobleme sowohl theoretisch als auch in der Praxis besser lösen. Wir haben unsere Ergebnisse genutzt, um eine Software zu entwickeln, die das General Routing Problem löst, und wir stellen Rechenergebnisse vor.

Abstract

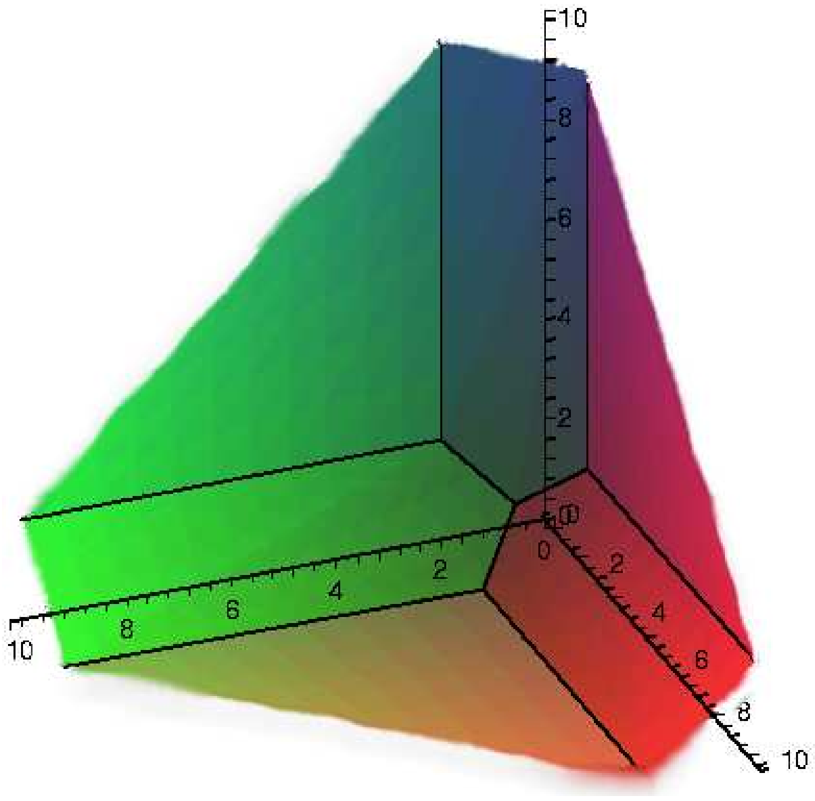
The General Routing Problem is an **NP**-hard combinatorial optimization problem defined on undirected graphs. It is a minor generalization of the better-known Rural Postman Problem (see, for example, Garey & Johnson 1979, [GJ79]), to which it is, both theoretically and practically, actually equivalent, although the names are historically different.

A widely successful approach to the practical solution of combinatorial optimization problems of this kind is to identify the set of feasible solutions for a particular instance of the problem with the extreme points of a convex polyhedron in an Euclidean vector space. A convex polyhedron is the intersection of a finite number of half spaces. Given a description of the polyhedron in terms of linear inequalities, standard methods of linear optimization can then be used to find the best solution quickly.

Using this approach, two main problems, one theoretical and one algorithmic, have to be attacked. Firstly, polyhedral investigations of the structure of the combinatorial optimization problem are required. Both the dimension of the ambient Euclidean spaces and the structure of the polyhedra themselves depend on the data which define the instance, in our case the graph. The goal is to find classes of linear inequalities which are necessary to describe the polyhedra. This branch of discrete optimization is commonly called polyhedral combinatorics. Secondly, the classes of linear inequalities have to be made usable algorithmically, which leads to the development of algorithms to solve the following problem: given a point in the ambient space, decide if it lies outside of the polyhedron defined by the instance, and if so, produce a hyperplane in the form of a linear inequality which separates the point from the polyhedron.

This thesis is dedicated to both the polyhedral combinatorics of the General Routing Problem and the algorithmic exploitation of polyhedral results. To the first issue, we contribute structural properties of the polyhedra and a number of formerly unknown classes of inequalities. To the algorithmic issue, we contribute both theoretically and practically improved separation algorithms. Finally, having applied our findings in the development of a piece of software which solves the General Routing Problem, we give computational results.

To Samuel



Preface

The General Routing Problem is an **NP**-hard combinatorial optimization problem defined on undirected graphs. It is a minor generalization of the better-known Rural Postman Problem (see, for example, Garey & Johnson [GJ79]), to which it is, both theoretically and practically, actually equivalent, although the names are historically different.

A widely successful approach to the practical solution of combinatorial optimization problems of this kind is to identify the set of feasible solutions for a particular instance of the problem with the extreme points of a convex polyhedron in $\mathbb{R}^m$. A convex polyhedron is the intersection of a finite number of half spaces. Given a description of the polyhedron in terms of linear inequalities, standard methods of linear optimization can then be used to find the best solution quickly.

Using this approach, two main problems, one theoretical and one algorithmic, have to be attacked. Firstly, polyhedral investigations of the structure of the combinatorial optimization problem are required. Both the dimension of the ambient spaces $\mathbb{R}^m$ and the structure of the polyhedra themselves depend on the data which define the instance, in our case the graph. The goal is to find classes of linear inequalities which are necessary to describe the polyhedra. This branch of discrete optimization is commonly called polyhedral combinatorics. Secondly, the classes of linear inequalities have to be made usable algorithmically, which leads to the development of algorithms to solve the following problem: given a point in the ambient space, decide if it lies outside of the polyhedron defined by the instance, and if so, produce a hyperplane in the form of a linear inequality which separates the point from the polyhedron.

This thesis is concerned with both the polyhedral combinatorics of the General Routing Problem, and the algorithmic exploitation of polyhedral results. To the first issue, our contributions include

- the solution of an open question about the structure of the polyhedra when there are only two so-called R-sets in Chapter 2,
- a structural result which allows extending known classes of inequalities in Chapter 3,
- the solution of an open question about lifting of facet-defining inequalities and the facet-defining property of known inequalities in Chapter 4, and
- the solution of an open question about the relationship of GTSP polyhedra (a subset of General Routing Problem polyhedra) with the polyhedra associated with the well-known Symmetric Traveling Salesman Problem, and deep insights into the relationship of the two kinds of polyhedra in Chapter 5.

An extended abstract of a subset of the results of Chapter 5, which was found in cooperation with M. Oswald, appeared in the proceedings of the 11th conference on Integer Programming and Combinatorial Optimization [ORT05]. Our contributions to the algorithmic issue include

- a new[1] algorithm for the so-called minimum odd cut problem, which, in practice, runs considerably faster than the known one,

[1]We acknowledge that the basic core idea of the algorithm was developed independently by Rizzi [Riz03].

- a new algorithm for the so-called blossom separation problem which improves the theoretical running time of the known algorithms for this problem in Chapter 7, and
- a number of results on polynomial time separation algorithms for some classes of inequalities in Chapter 8.

An extended abstract of an earlier version of the results on blossom separation, which were co-operatively developed with A. Letchford, appeared in the proceedings of the 10th conference on Integer Programming and Combinatorial Optimization [LRT04].

Finally, we also contribute to the practical solution of the General Routing Problem. We developed a so-called Branch-and-Cut software, which solves General Routing Problem instances, designed and implemented various heuristic separation algorithms described in Chapter 9, and we present computational results for General Routing Problem instances on graphs with up to and exceeding 2000 nodes in Chapters 11 and 12. The vast majority of the instances with up to more than 1000 nodes can be solved quickly by our software program.

For some obscure reason, theses are written in “we”-form, and this paragraph is the only exception to this rule in this thesis. I would like to thank my PhD supervisor Professor Gerhard Reinelt for financing my position and for his support. Adam Letchford and Marcus Oswald deserve particular thanks for their contributions to the results of Chapters 7 and 5, respectively. Thanks also to the physicist Tania Robens, who wrote code for the transformation of data files of problem instances and the extraction of statistical data. John Beasley, Angel Corberán, Richard Eglese, and Gerhard Reinelt contributed data files containing problem instances. Last but not least, thanks to my colleagues Dino Ahr, Cara Cocking (thanks for proof reading!), Marcus Oswald, Hanna Seitz (thanks for proofreading!), Chotiros Surapholchai (thanks for all the cookies!), and Klaus Wenger at the Discrete Optimization Group of the University of Heidelberg, and to the secretaries Catherine Proux-Wieland and Karin Tenschert. I dedicate this thesis, particularly my favourite Chapter 5, to my son, Samuel.

Contents

Chapter 0

Preliminaries and notation

In this chapter, we explain terminology and notation which we use, and we direct the reader to the mathematical preliminaries which we require. All symbols are also explained in the list of symbols at the end of this thesis.

0.1 Affine geometry and polyhedra

For sets $X, Y \subseteq \mathbb{R}^m$, we frequently write $X + Y$ for the set of all elements $x + y$ with $x \in X$ and $y \in Y$. We define $X - Y$ in the same fashion, and, in order to avoid confusion, we particularly direct the reader to the notation $X - X = \{x - y \mid x, y \in X\}$.

Let L be a real vector space. We recall the definition of an affine subspace, which is a set $A \subseteq L$ satisfying $\lambda a + \mu b \in A$ for all $a, b \in A$ and $\lambda, \mu \in \mathbb{R}$ with $\lambda + \mu = 1$. Let A be an affine subspace of a vector space. Then we say that the set $A - A$ is the *linear space defined by A*. Given any non-empty set $X \subseteq L$, we denote by $\operatorname{aff} X$ the affine hull of X, which is the smallest affine subspace containing X. We also call $\operatorname{aff} X - \operatorname{aff} X = \operatorname{lin}(X - X)$ the linear space defined by X, where $\operatorname{lin} Z$ denotes the linear hull of a set Z. The *dimension* $\dim X$ of a non-empty set X is the dimension of the linear space defined by X. In particular, the dimension of an affine subspace is the dimension of the linear space it defines. We agree that $\dim \emptyset = -1$.

linear space defined by A

$\operatorname{aff} X$

$\operatorname{lin} Z$

Let L be a real vector space with inner product $\langle \cdot \mid \cdot \rangle$. A *half space* is a set of the form $\{x \in L \mid \langle x \mid a \rangle \geq \alpha\}$, where $a \in L \setminus 0$ and $\alpha \in \mathbb{R}$. We say that the half space is *defined* by the inequality (a, α). For the notation (a, α) we allow $a = 0$. In $\mathbb{R}^m$, we use the notation xy for the standard inner product, thus (a, α) represents the inequality $ax \geq \alpha$. Note that in the general context, it is customary to use less-than-or-equal inequalities, but in our context, the greater-than-or-equal inequalities are more natural.

(a, α)

We denote by $\operatorname{conv} X$ the convex hull of a set X. A (convex) cone is a set which is closed with respect to addition and scalar multiplication with non-negative reals. The *conic hull* $\operatorname{cone} X$ of the set X is the smallest cone containing X. A (convex) *polyhedron* is an intersection of a finite number of half spaces. Equivalently, it is the Minkowski sum of a convex hull of a finite set X and the conic hull of a finite set Y, namely $\operatorname{conv} X + \operatorname{cone} Y$. A bounded polyhedron is called a *polytope.*

If P is a polyhedron, an inequality (a, α) is said to be *valid* for P, if $\langle a \mid x \rangle \geq \alpha$ for all $x \in P$. For non-zero a, this is equivalent to P being contained in the half space defined by (a, α). A *face* of a polyhedron is the intersection of P with the set $\{x \in L \mid \langle a \mid x \rangle = \alpha\}$, which is equal to the hyperplane in L defined by the equation if a is non-zero. A face of dimension zero is called a *vertex,* a face of dimension one is called an *edge.* By the *codimension* of a face F of a polyhedron P we mean the number $\operatorname{codim} F := \dim P - \dim F$. A face with codimension one is called a *facet,* a face with codimension at least one is called *proper.*

valid inequality

face

vertex, edge, facet

codimension

proper face

In Chapter 5, a slightly deeper understanding of convex polyhedra will be necessary, for which we refer to any text book on the subject (e.g., [Brø83, Grü03, Zie98]). We denote the face lattice

$\mathscr{L}(P)$, $\mathscr{F}(\cdot)$ of a polyhedron P by $\mathscr{L}(P)$, and for $x \in P$, we denote by $\mathscr{F}(x)$ the set of all facets of P which contain x.

0.1.1 Polyhedra of blocking type

We recall (see e.g. [Sch86]) that a polyhedron $P \subseteq \mathbb{R}^m$ is said to be of *blocking type,* if $P = \operatorname{conv} X + \mathbb{R}^m_+$, where $X \subseteq \mathbb{R}^m_+$ is a finite set. For a polyhedron $P \subseteq \mathbb{R}^m$, we define

$$B(P) := \{a \in \mathbb{R}^m_+ \mid ax \geq 1 \ \forall x \in P\}.$$

blocking polyhedron The set $B(P)$ is a polyhedron of blocking type. If P is of blocking type, then $B(P)$ is called the *blocking polyhedron* of P. P is of blocking type if and only if $B(B(P)) = P$ holds. Obviously, the non-negativity inequalities $x_j \geq 0$ define faces of P which are either empty or facets. We call *non-negativity facets* facets which are defined by non-negativity inequalities *non-negativity facets.* For convenience, in *trivial face* the following proposition, we say that a face of P or $B(P)$ is *trivial,* if it is proper and non-empty and only contained in non-negativity facets.

0.1.1 Proposition *The canonical mapping $B(P) \mapsto \mathbb{R}^m \times \mathbb{R}$: $a \mapsto (a, 1)$ which maps members of the blocking polyhedron to valid inequalities for P has the following properties.*

(a). For all $a \in \mathbb{R}^m \setminus \{0\}$, a is a vertex of $B(P)$ if and only if $(a, 1)$ defines a facet of P.

(b). Let $a \in \mathbb{R}^m \setminus \{0\}$ and $d \geq -1$. Then a is a relative interior point of a non-trivial face of $B(P)$ with codimension $d + 1$ if and only if $(a, 1)$ is valid and defines a face of dimension d of P.

(c). Let $\mathscr{L}$ denote the face lattice of P and let $\mathscr{N} \subseteq \mathscr{L}$ be the set of all trivial faces of P. Further let Let $\mathscr{L}^B$ denote the face lattice of $B(P)$ and let $\mathscr{N}^B \subseteq \mathscr{L}$ be the set of all trivial faces of $B(P)$. Then the posets $\mathscr{L} \setminus \mathscr{N}$ and $\mathscr{L}^B \setminus \mathscr{N}^B$ are anti-isomorphic. We denote this $\square^\sharp$ *anti-isomorphism by $\square^\sharp \colon F \mapsto F^\sharp$.*

(d). A face $F^\sharp$ of $B(P)$ is bounded if and only if F is not contained in a trivial face.

Proof. *(a)* See (b).

(b) For any $a \neq 0$, we define

$$Q := \{(x, 1) \mid x \in P \wedge ax = 1\}, \qquad \text{and} \qquad F := \{x \in P \mid ax = 1\}.$$

Clearly, $F = \emptyset$ is equivalent to $Q = \emptyset$, and, using $\operatorname{lin} \emptyset = \{0\}$, we have $\dim F = \dim \operatorname{lin} Q - 1$.

Let $a \neq 0$ be a relative interior point of a face $F^\sharp$ of codimension $d + 1$ of $B(P)$. Since $B(B(P)) = P$, if $F^\sharp$ is non-trivial, then for all $d \geq -1$ the linear space of *all* equations satisfied by $F^\sharp$ is generated by Q. On the other hand, clearly, the face defined by $(a, 1)$ is F.

For the other direction, let $(a, 1)$, with $a \neq 0$, be a valid inequality defining the face F with dimension d. Then a is contained as a relative interior point in a face $F^\sharp$ of P. Again $\operatorname{lin} Q$ is the space of equations satisfied by $F^\sharp$ and hence $\dim F^\sharp = \dim \operatorname{lin} Q$.

(c) Follows from (b), with $\emptyset^\sharp := B(P)$.

(d) Obvious. □

0.2 Graphs

For all the standard terminology related to graphs, we refer to any of the common text books, for example [BM76, Die00]. In this section we mainly explain the deviations from standard notation which are convenient and common in our context.

In this thesis we deal with two kinds of (abstract) graphs: simple graphs and loopless multi- *$V(G)$* graphs. For a graph G, we denote by $V(G)$ its node set. By $E(G)$ we denote the edge (multi)set, *$E(G)$* which is a sub-(multi-)set of the set of two-element subsets of $V(G)$. We will denote by uv an edge with end nodes u and v, i.e., in the case of (simple) graphs, uv is shorthand for $\{u, v\}$. If $uv \in E(G)$, we sometimes write $u \sim v$, meaning that u and v are adjacent. $u \sim v$

For a node set U, we follow [GR00] in denoting by $\partial(U)$ the *coboundary*[1] of U, i.e., the set of edges with precisely one end node in U. We abbreviate $\partial(\{u\})$ by $\partial(u)$. An index $\partial_H(\cdot)$ means that we take the edge set in the graph H. If $\emptyset \subsetneq U \subsetneq V(G)$, we usually say that $\partial(U)$ is a *cut* in the graph, but in Chapter 7, for convenience, we will say that $(U, V(G) \setminus U)$ is a cut. In any case, we will call U and $V(G) \setminus U$ the *shores* of the cut. We will also use the abbreviations $(U : V) := \partial(U) \cap \partial(V)$ and $(u : v) := (\{u\} : \{v\})$. $\partial(U)$ *cut* *shore* $(U : V)$, $(u : v)$

We denote by $G[U]$ the subgraph induced by the node set $U \subseteq V(G)$. $G[U]$

Recall that a *block* of G is a maximal connected subgraph which does not contain a cut-node of G. The reverse operation to decomposing G into blocks is the 1-*sum* operation. 1-*sum*

0.2.1 Deletion, contraction, shrinking, and identification

Let F be a set of edges of a graph G. We denote by $G \setminus F$ the graph which results if the edges in F are deleted from G. For a set $U \subseteq V(G)$, we denote by $G - U$ the graph which results if the nodes in G along with their incident edges are deleted from G. $G \setminus F$ $G - U$

We denote by G/F the graph obtained from G by contracting the edges in F, where, depending on the context, the result of the contraction is either a loopless multigraph, i.e., loops are deleted but parallel edges are kept, or a simple graph. The abbreviation $G/e := G/\{e\}$ is customary. Let H denote the subgraph of G induced by F. If H is connected, then G/F is called the graph obtained from G by *shrinking* H, or by shrinking $V(H)$, and it is also denoted by $G/V(H)$. In the general case, if $U_1, \dots, U_k$ are the node sets of the connected components of H, then the contraction operation replaces each set U_i by a node u_i, and we say that u_i is the node which results from shrinking U_i. G/F *shrinking*

We assume that the nodes (edges) of G which are not changed by the contraction form a subset of the node set (edge set) of G/F. For example, if $uv \in E(G)$, then $V(G/uv) \supseteq V(G) \setminus \{u, v\}$. In the case of loopless multigraphs, $E(G/uv) = E(G) \setminus (u : v)$, or, more generally, if H denotes the subgraph of G induced by F, then $E(G/F) = E(G) \setminus E(H)$.

For $U \subseteq V(G)$ we denote by G/U the graph obtained by *identifying the nodes in U to a single node.* In terms of contraction, this is the graph which results if for every 2-element subset $\{u, v\}$ of U with $uv \notin E(G)$, an edge uv is added to G, and then the set of all edges with both end nodes in U is contracted. In other words, $V(G/U) = V(G) \setminus U \cup \{u\}$ and the edge set of G/U is defined in the obvious way, depending on whether we speak of simple graphs or loopless multigraphs. We say that u is the node which results from identifying U. If $U_1, \dots, U_k$ are disjoint subsets of $V(G)$, then we can construct the graph which results from G by *identifying each set U_i to a single node u_i.* This means that the new graph has node set $\{u_1, \dots, u_k\} \cup V(G) \setminus \bigcup_i U_i$. *identifying*

0.2.2 Vector notations

We denote by $\mathbf{0}$ the all-zeroes vector, and by $\mathbf{1}$ the all-ones vector with appropriate index set. For a set F, we denote by χ^F the characteristic vector for the set F, i.e., $\chi^F_e = 1$, if $e \in F$, and $\chi^F_e = 0$, otherwise. We abbreviate $\chi^j := \chi^{\{j\}}$. $\mathbf{0}$, $\mathbf{1}$ χ^F, χ^e

Let $x\colon M \to X$, and $F \subseteq M$. we denote by $x_{|F}$ the restricted function $F \to X$. $x_{|F}$

Let G be a graph and X a set. We do not distinguish between a vector $x \in X^{E(G)}$ and a function $x\colon E(G) \to X : e \mapsto x_e$. Now let $x\colon E(G) \to \mathbb{R}$. For any set of edges $F \subseteq E(G)$, we abbreviate $x(F) := \sum_{f \in F} x_f$. We also let $E(x) := \{e \mid x_e \neq 0\}$, and if not otherwise stated, $G(x)$ denotes the spanning subgraph of G with edge set $E(x)$. The graph $G(x)$ is naturally equipped with edge the weights x_e, $e \in E(x)$. $x(F)$ $E(x)$ $G(x)$

For $x\colon E(G) \to \mathbb{R}$ and $F \subseteq E(G)$, we define the vector $x/F\colon E(G/F) \to \mathbb{R}$. If the contraction takes place in the context of loopless multigraphs, and $\bar{F}$ denotes the edge set of the subgraph H of G induced by F, then $x/F := x_{|E(G) \setminus \bar{F}}$. x/F

[1]Using ∂ instead of δ is somewhat counterintuitive, but we want to keep the symbol δ free of other purposes.

merging of edges

If the contraction takes place in the context of simple graphs, then, if $U_1, \dots, U_k$ are the node sets of the connected components of H, and u_i is the node which results from shrinking U_i, then

$$(x/F)_{vw} := \begin{cases} x(U_i : U_j) & \text{if } v = u_i \text{ and } w = u_j \\ x(v : U_j) & \text{if } v \notin \{u_1, \dots, u_k\} \text{ and } w = u_j \text{ and} \\ x_{vw} & \text{if } v, w \notin \{u_1, \dots, u_k\} \ . \end{cases}$$

We say that the edges $(U_i : U_j)$ (or $(v : U_j)$ resp.) are *merged.* The corresponding notation applies to node-identification.

0.2.3 Miscellany

$\operatorname{dist}(u, v)$

We denote by K_n the complete graph with n nodes. Given a graph G and a non-negative vector c, we denote by $\operatorname{dist}_{(G,c)}(u, v)$ the length of a shortest path in G with respect to c between u and v. We abbreviate $\operatorname{dist}_G(u, v) := \operatorname{dist}_{(G,\mathbf{1})}(u, v)$ and we omit the index G if it cannot be confused.

t-join, T-join

Let $t\colon V(G) \to \{0, 1\}$ with $\sum_v t(v) = 0 \bmod 2$. A *t-join* is a vector $y\colon E(G) \to \{0, 1\}$ with the property that $y(\partial(v)) = t(v) \bmod 2$ holds for all $v \in V(G)$ (see, e.g., [Cat92]). We will identify y with the edge set $J := \{e \mid y_e = 1\}$. In the area of combinatorial optimization, it is more common to call J a T-join with set of *odd* nodes $\{v \mid t(v) = 1\}$, or a T-join, where $T := \{v \mid t(v) = 1\}$.

0.3 Algorithms and complexity

We give a very brief and very informal introduction to algorithms and complexity, which is mainly adopted from [Coo] and [Sch03].

We call a problem *polynomially solvable,* or simply polynomial, if there exists a polynomial time algorithm for the problem, i.e., an algorithm whose running time in terms of number of "elementary" steps is bounded by a fixed polynomial in the size of the input.

Informally the class **P** is the class of decision problems which are polynomially solvable. **NP** is defined as the class of decision problems for which each input with positive answer has a polynomial-time checkable "certificate" of correctness of the answer [Sch03]. The class co**NP** consists of all decision problems, for which the complementary problem, i.e., the problem with positive and negative answer exchanged, is in **NP**. A decision problem P in **NP** is ***NP**-complete,* if every problem in $P' \in$ **NP** can be polynomially reduced to P, i.e., there exists a function T from the domain of P' to the domain of P, which can be computed in polynomial time, such that the answer to x' is positive if and only if the answer to $T(x')$ is positive.

An optimization problem (or, to be precise, a minimization problem) is a problem of the form $\min\{f(x) \mid x \in X\}$, where X is a set derived from the input of the problem, and f is a rational-valued function on X. It can usually be reduced to the following decision problem: "Given a $r \in \mathbb{Q}$, is there an $x \in X$ with $f(x) \leq r$?" Obviously, the decision problem can be polynomially reduced to the optimization problem. On the other hand, if an upper bound β on the size of the minimum value of f is known such that β is bounded by a polynomial in the input size, then, by binary search, the optimization problem can be polynomially reduced to the decision problem.

An optimization problem P is called ***NP**-hard,* if every decision problem in **NP** can be reduced to P in the manner just described.

0.4 Combinatorial optimization

We repeat some facts from the area of combinatorial optimization.

0.4.1 Minimum cuts and cactus representation

submodular function

Given a set V, a function $\mathbf{2}^V \to \mathbb{Q}$ is called *submodular,* if for each $U, W \subseteq V$ we have $f(U \cap W) +$

$f(U \cup W) \leq f(U) + f(W)$. It can easily be seen that, if G is a graph and $c \in \mathbb{Q}_+^{E(G)}$, then the function

$$c(\boldsymbol{.})\colon \mathbf{2}^{V(G)} \to \mathbb{Q}\colon U \mapsto c(U) := c(\partial(U))$$

is submodular. The minimum value which $c(\boldsymbol{.})$ attains over $\mathbf{2}^{V(G)} \setminus \{\emptyset, V(G)\}$ is denoted by $\lambda_c(G)$, and for a set U on which the minimum is attained, $\partial(U)$ is called a *c-minimum cut,* or just *minimum cut.* In Chapter 7, we will call the pair $(U, V(G) \setminus U)$ a (minimum) cut, instead of the edge set. If G is not connected, then $\lambda_c(G) = 0$, and for the case that G consists of a single node, we agree on $\lambda_c(G) = \infty$.

circular partition

A *circular partition* $A_0, \ldots, A_K$ is a partition of $V(G)$, which satisfies $c(A_i : A_{i+1}) = \lambda_c(G)/2$ for each i, where $i+1$ is computed modulo K. All circular partitions of can be retrieved from the so-called cactus representation of all minimum cuts of G. A *cactus* is a graph in which every block is a K_2 or a circle. The *cactus representation of all minimum cuts* of a graph is a cactus $\mathcal{K}$ together with a mapping $\pi\colon V(G) \to V(\mathcal{K})$ with the property that every cut in $\mathcal{K}$ consisting of one or two edges induces, by the application of π^{-1} to the node sets of the shores, a c-minimum cut in G. A cactus representation of all minimum cuts exists for all graphs (with at least two nodes) and all $c\colon E(G) \to \mathbb{Q}_+$ [NK94]. It can be computed in time in time $O(nm \log({n^2}/{m}))$ [Fle99].

cactus representation

Let s and t be two distinct nodes of G. An (s,t)-cut is a cut $\partial(S)$, separating the nodes s and t, i.e., with $s \in S$ and $t \in V(G) \setminus S$, and a minimum (s,t)-cut is an (s,t)-cut which minimizes the submodular function $c(\boldsymbol{.})$ subject to this condition. By $\lambda_c(G,s,t)$, we denote the value of a c-minimum (s,t)-cut in G, where we omit G and c if possible. For Chapter 7, we assume that the reader is familiar with the basics of maximum (s,t)-flows, minimum (s,t)-cuts, and the residual network of a flow. When we talk of an (s,t)-cut $(X, V(G) \setminus X)$, we mean that $s \in X$ and $t \in V(G) \setminus X$.

We recall some essential facts about the algorithm of Hao & Orlin [HO92] for computing a minimum cut in a graph. For $j = 1, \ldots, n-1$, where $n := |V(G)|$, the algorithm computes a minimum (s_j, t_j)-cut in the graph $G_j := G/S_j$, where $S_1 \subseteq \ldots \subseteq S_{n-1} \subseteq V(G)$, $|S_j| = j$, and s_j is the node of G_j which results from identifying S_j to a single node. The minimum over the computed minimum (s_j, t_j)-cut values equals $\lambda(G)$. While the idea is simple and goes back to [PR90b], the contribution of [HO92] lay in showing how all the minimum (s_j, t_j)-computations can be performed in total running time of $O(nm \log {n^2}/{m})$.

0.4.2 Cutting-planes, separation, and Branch-and-Cut

A successful method for solving combinatorial optimization problems is to formulate them as an *Integer-Linear-Programming (IP-) problem*

$$\begin{aligned} \min cx&, \text{ subject to} \\ Ax &\geq b \\ x &\text{ integer.} \end{aligned} \tag{0.1}$$

and then use cutting-plane and Branch-and-Cut techniques, which we will now describe.

By omitting the condition that x must be integer from the system (0.1), one obtains a so-called *LP-relaxation* of the IP. Its solution by, for example, the Simplex method, gives a lower bound for the optimal solution of the combinatorial optimization problem. We assume a certain low-level familiarity with the Simplex method. The LP-relaxation is then successively strengthened by adding more inequalities to the system $Ax \geq b$. To be precise, suppose that x^* is a vector which satisfies $Ax^* \geq b$. The problem of deciding whether there exists an inequality (a, α) separating x^* from the set $P := \operatorname{conv}\{x \in \mathbb{Z}^m \mid Ax \geq b\}$, and if so, producing one, is called the *separation problem* for P. It is a famous theorem in combinatorial optimization that, under some technical conditions, the combinatorial optimization problem can be solved in polynomial time if and only if the separation problem is polynomial [GLS93]. See Algorithm 0.1 for a generic cutting-plane procedure based on the Simplex method (for the sake of brevity, we excluded the case that the LP is unbounded). This is the way how cutting-plane approaches are usually used in practice. An

LP-relaxation

separation

iteration of the loop 1–8 is called an iteration of the cutting-plane algorithm. Theoretically, it is not guaranteed that the algorithm terminates after a polynomial number of iterations (or even a finite number, unless conditions on the (a, β) are demanded).

Algorithm 0.1 Cutting-plane algorithm

Input:

System of linear inequalities (A, b) and cost vector c such that $x \mapsto cx$ is bounded on $\{x \mid Ax \geq b\}$.

Output:

Solution to $\min cx$ subject to $Ax \geq b$, x integer, if such an x exists.

```
1: **Loop**
2:     Solve the LP-relaxation $\min cx$ subject to $Ax \geq b$ using the dual simplex method.
3:     If the LP is infeasible, output "no lattice point", **Stop**.
4:     Let $x^*$ denote the LP solution.
5:     *Separation routine:* Produce inequalities $(a, \beta)$ which are valid for
       $P := \operatorname{conv}\{x \in \mathbb{Z}^m \mid Ax \geq b\}$, but violated by $x^*$, i.e., $ax^* < \beta$.
6:     If none could be found, output "$x^*$ is an optimal solution", **Stop**
7:     Add the inequalities $(a, \beta)$ to the system $(A, b)$.
8: **End loop**
```

From the above mentioned polynomial equivalence of separation and optimization it follows that, for an **NP**-hard optimization problem, the running time expense required for the exact resolution of the separation problem may not be feasible.

Although the cutting-plane algorithm could theoretically be used to solve an IP, in practice, the method is combined with branch-and-bound techniques to form the so-called *Branch-and-Cut (B&C)* approach, cf. [PR87]. Fig. 1 displays[2] the core algorithm. The algorithm maintains a rooted tree of subproblems, called nodes. The B&C-process starts with one (root) subproblem. In the cutting-plane phase, the loop between "solve LP" and "new constraints", the local lower bounds (llb) for the subproblems are determined and increased by cutting-plane generation ("separate"). If "separate" fails to find valid inequalities violated by the current LP-solution, or if the local lower bound does not increase for a number of iterations of the cutting plane phase ("tailing off"), then the subproblem is split into two new subproblems ("branch"), in the sense that the union of the sets of integer solutions in both subproblems equals the set of integer solutions in the split subproblem. Then a subproblem in the tree is selected ("select") for further treatment. Whenever during the cutting plane phase an LP turns out to be infeasible, a local lower bound (llb) of a subproblem exceeds the upper bound (ub), or if the subproblem is solved to optimality ("feasible") the subproblem can be pruned from the tree. The approach also incorporates the idea of exploiting the solution of the LP-relaxation to find integer feasible solution, by which the upper bounds can be improved, i.e., reduced ("feasible" and "exploit LP").

Branch-and-Cut, B&C

branching

The process finishes as soon as there is no remaining node in the tree ("tree empty").

A global lower bound (glb) can be defined as the minimum of all local lower bounds of subproblems in the tree. The number

$$\frac{\text{ub} - \text{glb}}{\text{glb}}$$

is then called the *gap.* It is common praxis to terminate the B&C-process after a certain time limit has been reached, and measure its success by the gap it achieved.

gap

In the separation phase, it is customary to invoke a number of algorithms which are dedicated to finding a violated valid inequality of a certain class. We call this a *separation algorithm for a class of inequalities,* and we speak of exact separation, if the algorithm finds a violated inequality iff one exists. It is customary to use *separation heuristics,* which may find violated inequalities of the respective class, but may also fail to do so, even if the solution of the LP-relaxation violates inequalities of the class.

exact/heuristic separation

[2]Thanks to Chotiros Surapholchai for the drawing, which is adopted from [JT98].

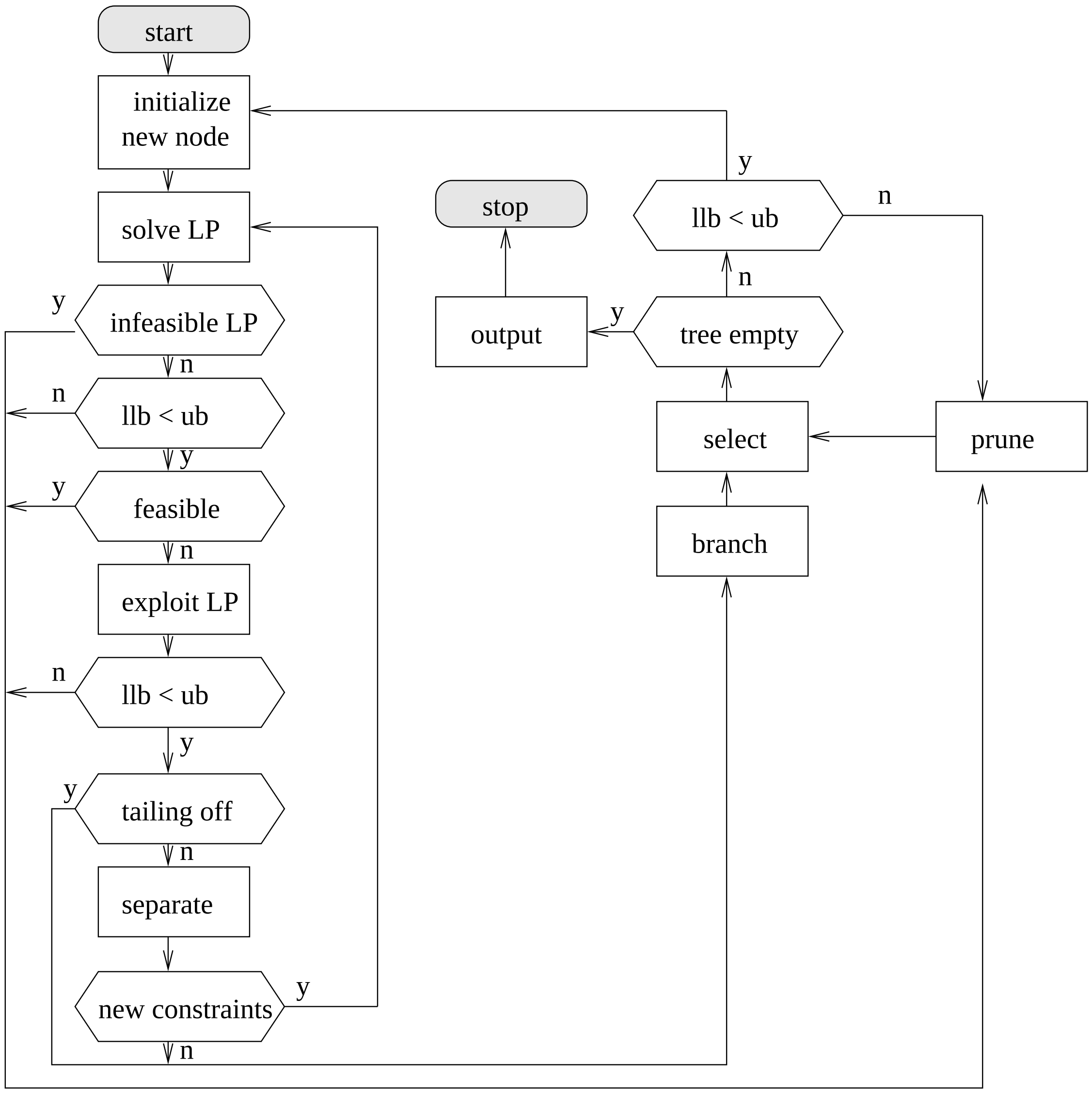


Figure 1: Branch-and-Cut flowchart

pool separation

To avoid unnecessary growth of the number of rows in the LP, rows are removed from time to time. For example, a row might be removed whenever it is not contained in the dual basis, or if the respective inequality (a, α) has a negative slack $\alpha - ax^*$, where x^* is the solution of the LP. The removed rows are stored in a so-called pool, and after a number of iterations of the cutting-plane phase, the inequalities in the pool can be checked, if they have become violated again after they were removed from the LP. This checking is called *pool separation.*

0.4.3 The Traveling Salesman Problem

STSP

The *Symmetric Traveling Salesman Problem, STSP,* is the **NP**-hard optimization problem which asks for a minimum cost Hamiltonian cycle in a complete graph K_n with edge weights $c\colon E(K_n) \to \mathbb{Q}_+$. A successful Branch-and-Cut approach to solve the STSP is based on the Integer-Programming-formulation usually attributed to [DFJ54]:

$$\begin{aligned} &\min cx \\ &x(\partial(u)) = 2 \text{ for all } u \in V(K_n) && \text{(0.2a)} \\ &x(\partial(U)) \geq 2 \text{ for all } \emptyset \subsetneq U \subsetneq V(K_n) && \text{(0.2b)} \\ &x \geq \mathbf{0} && \text{(0.2c)} \\ &x \text{ integer.} \end{aligned}$$

The equations (0.2a) are called *degree equations,* the inequalities (0.2b) are refereed to as subtour elimination constraints. The LP-relaxation can be strengthened by adding the so-called *comb inequalities,*

$$x(\partial(H)) + \sum_{j=1}^{t} x(\partial(T)) \geq 3t + 1,$$

where $\emptyset \neq H \subsetneq V(K_n)$, and $\emptyset \neq T_j \subsetneq V(K_n)$, for $j = 1, \ldots, t$, the T_j are pairwise disjoint and $T_j \cap H, T_j \setminus H \neq \emptyset$. In order for the inequality to make sense, t must be greater than or equal to three an odd number.

STSP(n)

We denote by STSP(n) the *Symmetric Traveling Salesman Polytope* on n nodes, which is defined as the convex hull of the integer points satisfying (0.2), i.e., the incidence vectors of edge sets of Hamiltonian cycles of the complete graph K_n.

Chapter 1

Rural Postman Problem and General Routing Problem

1.1 Definition

1.1.1 Classical definition

Given a connected loopless multigraph G, a vector $c \in \mathbb{R}_+^{E(G)}$ of non-negative edge costs, a set of *required nodes* $V_R \subseteq V(G)$ and a set of *required edges* $E_R \subseteq E(G)$, the *General Routing Problem (GRP)* consists of finding a closed walk in G containing the required nodes and edges, such that the sum of the edge costs is minimized [Orl74, LR76].

required edges E_R

The General Routing Problem includes as a special case the *Rural Postman Problem (RPP)*, namely if $V_R = \emptyset$. However, the GRP and RPP are not essentially different, and a GRP-instance can be "modeled" as an RPP-instance by simply adding a new node and a single edge for each node in V_R which is not adjacent to a node in E_R. The definition of the General Routing Problem is more natural than the original definition of the Rural Postman Problem for a number of reasons. For example, the definition of the GRP is closed to block-decompositions: If a GRP-instance is defined on a graph which is not 2-connected, then the solution of the instance is equivalent to solving the GRP on each block. For the RPP, the relationship is an unnecessary bit more complicated, since the instances on the blocks may be RPPs, but they may also be genuine GRPs. We would like to advocate at this point, to amend the definition of the Rural Postman Problem to match that of the General Routing Problem.

RPP

If $E_R = \emptyset$ and $V_R = V(G)$, we obtain the *Graphical Traveling Salesman Problem (GTSP)*. The GRP, RPP, and GTSP are **NP**-hard combinatorial optimization problems [LR76, CFN85].

GTSP

There exists a preprocessing procedure for the GRP which allows us to assume without loss of generality that $V_R = V(G)$ [CCCM81]. The modification is straight forward, and we do not repeat it here. In this thesis we exclusively consider the case $V_R = V(G)$.

A *semitour* [CS94, CS98] is a vector $x \in \mathbb{Z}_+^{E(G)}$, with the property that $x + \chi^{E_R}$ corresponds to the edge multiset of a spanning closed walk in G (which is equivalent to being a feasible solution to the GRP because $V_R = V(G)$). Clearly, finding a minimum cost feasible solution to the GRP is equivalent to finding a semitour x with minimum cost $cx := \sum_{e \in E(G)} c_e x_e$.

To characterize the set of semitours, we introduce some more terminology. First we define the *parity* of a node $u \in V(G)$ as

$$\mathbf{t}(u) := |\partial(u) \cap E_R| \bmod 2.$$

Hence, $\mathbf{t}(u) = 1$ if there is an odd number of required edges incident to u and $\mathbf{t}(u) = 0$ if the number is even. Further, let $\mathcal{C} = \{C_1, \dots, C_k\}$ denote the node sets of the connected components of the spanning subgraph of G with edge set E_R. The sets C_i form a partition of the node set $V(G)$.

The following system characterizes the set of semitours [CS94, CS98]

$$x(\partial(u)) = \mathbf{t}(u) \mod 2 \qquad \text{for all } u \in V(G) \tag{1.1a}$$
$$x(\partial(S)) \geq 2 \qquad \text{for all } S = \bigcup\nolimits_{C\in\mathcal{S}} C,\ \emptyset \neq \mathcal{S} \subsetneq \mathcal{C} \tag{1.1b}$$
$$x_e \geq 0 \qquad \text{for all } e \in E(G) \tag{1.1c}$$
$$x \in \mathbb{Z}_+^{E(G)}, \tag{1.1d}$$

The (modular) equations (1.1a) are called *parity constraints* and the inequalities (1.1b) are called *connectivity inequalities.*

1.1.2 Our more general definition

R-set partition $\mathcal{C}$ *parity function* $\mathbf{t}$ *GRP-structure,* $\Gamma = (G, \mathcal{C}, \mathbf{t})$

For the characterization of semitours in (1.1), we only need the *R-set partition* $\mathcal{C}$ and the *parity function* $\mathbf{t}$, but not the required edges. In fact it will prove useful to restrict the attention to some axioms on the triple $(G, \mathcal{C}, \mathbf{t})$, and forget about required edges in most places. We call a triple $\Gamma := (G, \mathcal{C}, \mathbf{t})$ a *GRP-structure,* if

1. G is a connected loopless multigraph, $\mathcal{C}$ is a partition of $V(G)$, and $\mathbf{t}$ is a mapping to the 2-element group $\{0, 1\}$
$$\mathbf{t}\colon V(G) \to \{0, 1\},$$
2. for each $C \in \mathcal{C}$, the induced subgraph $G[C]$ is connected,
3. for each $C \in \mathcal{C}$, the set C is *R-even,*

$\mathbf{t}(U)$

where we define the *R-parity,* or just *parity,* $\mathbf{t}(U)$ *of a set* U to be

$$\mathbf{t}(U) := \sum_{u\in U} \mathbf{t}(u) \mod 2,$$

R-even/odd set *R-sets* *R-isolated* $G^{\mathrm{m}}_{\mathcal{C}}$,$G^{\mathrm{s}}_{\mathcal{C}}$ *R-external* *R-internal* E_{int} *semitour*

and we say that a set is *R-even* (*R-odd*) if its parity is zero (one).

The sets $C \in \mathcal{C}$ are called *R-sets.* If an R-set consists of a single node, then we say that this node is *R-isolated.* We define the loopless multigraph $G^{\mathrm{m}}_{\mathcal{C}}$, which results from shrinking each R-set to a single node. The node of $G^{\mathrm{m}}_{\mathcal{C}}$ which results from the R-set C is again denoted by C, i.e., we assume that the node set of $G^{\mathrm{m}}_{\mathcal{C}}$ is $\mathcal{C}$. The *simple* graph which results from shrinking each R-set to a node is denoted by $G^{\mathrm{s}}_{\mathcal{C}}$. The edge set of $G^{\mathrm{m}}_{\mathcal{C}}$ is a subset of $E(G)$, consisting of all edges which have their end nodes in different R-sets. We call these edges *R-external,* and we call edges of G with both end nodes in the same R-set *R-internal.* The set of R-internal edges is denoted by $E_{\mathrm{int}}(\Gamma)$.

If Γ is a GRP-structure, a solution x to (1.1) is again called a *semitour.* We denote the set of all semitours, i.e., the set of all solutions to (1.1), by $\mathscr{S}(\Gamma)$.

Note that, with this definition, we can assume that G is a simple graph, if the context requires this. If the graph on which the GRP instance is defined is not simple, we can delete from each set of parallel edges all but the edge with least cost, even if that involves the deletion of required edges. This is possible because we define the GRP-structure without using required edges. Thus every GRP-instance defines a GRP-structure with a simple graph and every GRP-structure with a simple graph can be derived from at least one GRP-instance, but duplication of edges may be necessary. There may be many different sets of required edges which define the same GRP-structure. However, in the context of lifting in Chapter 4, we will need to maintain parallel edges, because the coefficients of an inequality may differ on parallel edges.

1.2 A short summary of known results

We briefly summarize what is known about the Rural Postman Problem and General Routing Problem in terms of polyhedra and cutting-plane algorithms. Some papers were dedicated to the study of the polyhedron which is the convex hull of all semitours, $\operatorname{conv}\mathscr{S}(\Gamma)$. If $E_R = \emptyset$, then the semitours are just the feasible solutions of the GTSP, so the GRP polyhedron is a generalization GTSP polyhedron, which was comparatively well understood [CFN85, NR88, NR91, NR93].

In [CS94] the RPP polyhedron $\operatorname{conv}\mathscr{S}(\Gamma)$ was first introduced, and basic classes of facet-defining inequalities, namely connectivity and R-odd cut inequalities, were presented. This paper also introduced the class of the so-called KC-inequalities, which define facets of the polyhedron under certain mild conditions. In [Let96], the so-called path inequalities [CFN85] for the GTSP were transferred to the GRP, where they have been called path-bridge inequalities. Further a polynomial time separation routine for a small subclass was introduced. The paper [CS98] studied the relationship between the GTSP and the GRP polyhedra, and it introduced the class of the facet-defining so-called honeycomb inequalities. It also transferred results of [NR91] about composition of facet-defining inequalities to the GRP. [Let99] proposed to view the GRP-polyhedron as a face of the GTSP-polyhedron, and helped to understand facets of the GRP in terms of facets of the GTSP. The PhD thesis of J. Sanchis [San90] is about RPP polyhedra and separation issues, and the PhD thesis of A. Letchford [Let97] deals in parts with the RPP and GRP polyhedra.

A new direction in the research on the RPP was ushered in by Ghiani & Laporte [GL00], who were the first to give an IP-formulation which only uses variables for edges of the graph, but it requires that a certain number of edges must be duplicated beforehand. They introduced the cocircuit inequalities for their polytope, but they failed to prove sufficient conditions for the facet-defining property of all basic valid inequalities except for the non-negativity and upper bound inequalities $0 \leq x_e \leq 1$. In [GL00], computational results for a B&C-algorithm are given, which uses the basic set of classes of valid inequalities proposed in [GL00].

Corberán, Letchford and Sanchis [CLS01] have described separation heuristics for KC-, honeycomb, and so-called regular path-bridge inequalities, and they gave computational results for a cutting-plane algorithm for the GRP which does not use the cocircuit inequalities of [GL00].

A line of research which has not been taken up by other researchers was proposed by Fernandez et al. [FMGO03]. Their approach requires to work on an almost complete graph, in which every edge is assigned the cost of the shortest path with respect to the original costs between the end nodes. Thus one reason why their results have not found much response is due to the loss of structural information, and the increase of running time required for the solution of the Linear Programs, both of which are consequences of the modifications on the graph. Another reason for the fact that the paper [FMGO03] has not been received very well is perhaps that the core of the IP-formulation it proposed is a flow-formulation which requires a number of additional flow inequalities and constraints. However, it is quite easy to see that if the connectivity inequalities are separated, then these additional variables and constraints are redundant in the sense that they do not improve the value of the LP-relaxation. In the cutting-plane algorithm which was proposed in [FMGO03], both the connectivity inequalities and the flow variables are used.

We finally state an easy result of [Fre79] about the complexity of the RPP. It is easy to see that, if $t_{G_\mathfrak{C}}$ is the number of spanning trees in $G^{\mathrm{m}}_{\mathfrak{C}}$, then the GRP can be solved by computing $t_{G_\mathfrak{C}}$ T-joins in the graph G. If $k := |\mathfrak{C}|$ and $l := |E(G^{\mathrm{m}}_{\mathfrak{C}})|$, then $t_{G_\mathfrak{C}} \leq l^{k-1}k^{k-2}$. Thus, there exists an exact algorithm for the RPP which is exponential in the number of R-sets, but which grows only polynomially with the number of nodes and edges of G if the number of R-sets is fixed. This fact must be kept in mind when considering computational results for the GRP. The instances for which [GL00] gives computational results have up to 350 nodes, but much fewer than 100 R-sets. In [CLS01], graphs with up to about 200 nodes and 100 R-sets are considered. As a rule of thump, the instances which are more difficult for cutting plane and Branch-and-Cut algorithms are more frequently found among those which have a relatively high number of R-sets. (In the last two chapters of this thesis, we will present computational results for instances with up to 2500 nodes and 2000 R-sets.)

Part I

Polyhedra

Chapter 2

Polyhedra associated with the GRP

We refer to a vector $b \in \big(\mathbb{Z}_+^* \cup \{\infty\}\big)^{E(G)}$ as a *bound vector*. For a GRP-structure $\varGamma$ and a bound vector b we define *bound vector*

$$\begin{aligned} \mathscr{S}(\varGamma, b) \quad &:= \Big\{x \in \mathbb{Z}_+^{E(G)} \mid x \text{ satisfies (1.1) and } x \le b\Big\} \\ \mathrm{GRP}(\varGamma, b) &:= \operatorname{conv}\big(\mathscr{S}(\varGamma, b)\big) \end{aligned}$$

Part I of this thesis will be concerned with these polyhedra. The unbounded polyhedron, which we denote by $\mathrm{GRP}(\varGamma, \infty)$, was introduced in [CS94]. Two further polytopes described in 2.2.1 below were introduced by Ghiani & Laporte [GL00], who showed that with certain conditions on b, which depend on the cost function, optimizing over $\mathrm{GRP}(\varGamma, b)$ gives an optimal solution for the General Routing Problem. In this chapter we give an introduction to basic properties of the polyhedra. In the first section we survey the unbounded polyhedron, namely structural properties and known classes of facet-defining inequalities. The second section explains the result of Ghiani & Laporte [GL00], to which we contribute the fact that facet-defining inequalities of $\mathrm{GRP}(\varGamma, b)$ with b finite do not necessarily have "configuration" form in 2.2.2. In the Section 2.3 we give results on polyhedra for GRP-structures with one or two R-sets, in particular we present some new classes of facet-defining inequalities for the unbounded polyhedron in this case.

In the last section of this chapter, we introduce a new polytope for the GRP which is the convex hull of a subset of the set $\mathscr{S}(\varGamma, \mathbf{b}^T)$ of semitours used in [GL00]. What distinguishes this polytope from the polyhedra $\mathrm{GRP}(\varGamma, b)$ is that the number of integer solutions depends only on $|V(G)|$ and $|E(G)|$, and they can easily be enumerated.

2.1 The unbounded polyhedron

In this section we deal with the well-studied polyhedron $\mathrm{GRP}(\varGamma, \infty)$ [CS94, San90, Let96, CS98, Let97, Let99]. We repeat only the facts which are necessary to understand the remainder of this thesis. For a more complete summary of known results we refer to [EL00]. In 2.3.1 we contribute some new classes of facets for $\mathrm{GRP}(\varGamma, \infty)$, which refer to the easiest non-trivial case, namely if there are precisely two R-sets. For one R-set the polyhedron is the T-join polyhedron, which is known (see Section 2.3 below).

It is easy to see that $\mathrm{GRP}(\varGamma, \infty)$ is of blocking type. As a consequence (see 0.1.1), a non-negativity inequality (1.1c) $x_e \ge 0$ defines a facet of $\mathrm{GRP}(\varGamma, \infty)$, iff there exists an $x \in \mathrm{GRP}(\varGamma, \infty)$ with $x_e = 0$, which is the case if e is not a cut-edge of G or, if it is, e is R-internal and the cut is even [CS94].

2.1.1 Configurations and 0-node lifting

We start with some structural results about $\mathrm{GRP}(\Gamma,\infty)$, which are discussed in [CS94]. Let $\Gamma = (G, \mathcal{C}, \mathbf{t})$ be a GRP-structure. Some definitions are necessary.

generalized triangle inequality

2.1.1 Definition We say that a vector $a \in \mathbb{R}_+^{E(G)}$ satisfies the *generalized triangle inequality,* if for each $uv \in E(G)$ the a-lengths the u,v-paths are at most a_{uv}, i.e., the path u,e,v is a shortest u,v-path with respect to a.

The following result is quite easy to see.

2.1.2 Proposition ([NR91, CS94, CS98]) *If a valid inequality (a,α) for* $\mathrm{GRP}(\Gamma,\infty)$ *does not satisfy the generalized triangle inequality, then it is dominated by a non-negativity inequality.*

configuration

configuration inequality

2.1.3 Definition A *configuration* [CS94] is a pair $(\mathcal{U}, A)$ consisting of a partition $\mathcal{U}$ of $V(G)$ together with a symmetric $\mathcal{U}\times\mathcal{U}$-matrix A such that $A_{UV} = 0$ if and only if $U = V$. We say that an inequality (a,α) is a *configuration inequality,* if there exists a configuration $(\mathcal{U}, A)$ such that for all $uv \in E(G)$

$$a_{uv} = A_{UV} \text{ whenever } u \in U \text{ and } v \in V.$$

An immediate consequence of Proposition 2.1.2 is the following.

2.1.4 Corollary ([NR91, CS94, CS98]) *A valid inequality which is not a configuration inequality is dominated by a non-negativity inequality.*

We show that contraction preserves facet-defining property of configuration inequalities. Before we can give the exact statement, we have to make precise how the R-set partition is defined for the contracted graph.

$\mathcal{D}/f$

$\mathcal{D}/F$

2.1.5 Definition Let $\mathcal{D}$ be a partition of the node set of a graph G, and let $uv = f \subseteq E(G)$. We let $\mathcal{D}/f$ denote the partition of $V(G/f)$ which results from $\mathcal{D}$ by taking the union of the sets containing u and v and replacing in this set the nodes u and v by the new node v_f which results from contracting f. For an edge set $F \subseteq E(G)$, we define $\mathcal{D}/F$ by successively contracting the edges in F.

$(a^\bullet,\alpha)$, $\Gamma^\bullet$

Now let (a,α) be a configuration inequality for $\mathrm{GRP}(\Gamma,\infty)$ with configuration $(\mathcal{U}, A)$, and consider the simple graph $G^\bullet$ which results from contracting the set E_0 of all edges e with $a_e = 0$, or, more precisely, $V(G^\bullet) = \mathcal{U}$ and $E(G^\bullet) = \{UV \mid (U:V) \neq \emptyset\}$. We emphasize the fact that for polyhedra without bounds, $\mathrm{GRP}(\Gamma,\infty)$, we contract to a *simple* graph. Let $\mathbf{t}^\bullet(U) = \mathbf{t}(U)$ for $U \in V(G^\bullet)$, and $\mathcal{C}^\bullet := \mathcal{C}/E_0$. Denote $\Gamma^\bullet := (G^\bullet, \mathcal{C}^\bullet, \mathbf{t}^\bullet)$ and, for all U, V, let $a^\bullet_{UV} := A_{UV}$. The proof of the following fact is elementary.

2.1.6 Lemma *If (a,α) defines a facet of* $\mathrm{GRP}(\Gamma,\infty)$*, then the inequality $(a^\bullet,\alpha)$ is facet-defining for* $\mathrm{GRP}(\Gamma^\bullet,\infty)$. □

Γ°, (a°,α)

The reverse process to the contraction of edges with zero coefficient is the so-called 0-node lifting. It is a common approach for proving facet-defining property for a class of inequalities: prove it for small graphs G and then use 0-node lifting, which means that validity and facet-defining property are inherited to graphs G° which can be contracted to G. To be precise, let $\Gamma^\circ = (G^\circ, \mathcal{C}^\circ, \mathbf{t}^\circ)$ be a GRP-structure and b° a bound vector. Let a partition $\mathcal{U}$ of the node set of G° be given such that the induced subgraphs $G^\circ[U]$, $U \in \mathcal{U}$, are connected. Let G denote the simple graph which results if each of the sets $U \in \mathcal{U}$ is shrunk to a single node. Further, abbreviate $\mathcal{C} := \mathcal{C}^\circ/F$, with $F := \bigcup_U E(U)$, and $b := b^\circ/F$. Note that the bounds on the merged edges are added (cf. the definition in 0.2.2 on page4). Suppose that $\Gamma = (G, \mathcal{C}, \mathbf{t})$ is again a GRP-structure which satisfies $\mathbf{t}(U) = \mathbf{t}^\circ(U) = \sum_{u \in U} \mathbf{t}^\circ(u) \bmod 2$ for all $U \in V(G) = \mathfrak{U}$. Finally, let (a, α) be a valid inequality for $\mathrm{GRP}(\varGamma, b)$. Define the coefficients a° on $E(G^\circ)$, by

$$a^\circ_e = \begin{cases} a_{UV} & \text{if there are } U, V \in \mathfrak{U} \text{ with } e \in (U : V) \\ 0 & \text{if } e \in E(U) \text{ for an } U \in \mathfrak{U}. \end{cases}$$

It is easy to see that the inequality (a°, α) is valid for $\mathrm{GRP}(\varGamma^\circ, b^\circ)$. We call it the inequality obtained by *0-node lifting* (a, α). The following fact is both easy and well-known.

2.1.7 Proposition ([NR91, CS94]) *If (a, α) is facet-defining for $\mathrm{GRP}(\varGamma, \infty)$, then (a°, α) is facet-defining for $\mathrm{GRP}(\varGamma^\circ, \infty)$.*

2.1.2 Known classes of facet-defining inequalities

In this section we review the most prominent constructions of facet-defining inequalities for the unbounded polyhedron $\mathrm{GRP}(\varGamma, \infty)$.

Connectivity and R-odd-cut inequalities

In [CS94], the most basic classes of facet-defining inequalities for the GRP were introduced: the non-negativity inequalities, the connectivity inequalities (1.1b) and the so-called R-odd-cut inequalities. Using 0-node lifting (see 2.1.1), it is easy to see that connectivity inequality $x(\partial(S)) \geq 2$ defines facets of $\mathrm{GRP}(\varGamma, \infty)$, if and only if the cut is a cocircuit, i.e., minimal with respect to inclusion [CS94]. This is equivalent to both shores of the cut being connected. To define R-odd-cut inequalities, we introduce some terminology.

2.1.8 Definition A cut $\partial(U)$ is called *R-odd* (*R-even*), if U is an R-odd (R-even) set. *R-odd cut*

Using 0-node lifting (see 2.1.1), it is easy to see that for an R-odd cut $\partial(U)$ the *R-odd-cut inequality* *R-odd-cut inequality*

$$x(\partial(U)) \geq 1 \tag{2.1}$$

is valid for $\mathrm{GRP}(\varGamma, \infty)$, and that it defines a facet of the polyhedron if and only if the cut is a cocircuit [CS94]. We now come to more subtle inequalities.

Path-bridge (PB-)inequalities

The path inequalities known from the TSP [CFN85] were introduced into the context of the GRP by Letchford [Let96] under the name of *path-bridge inequalities,* or *PB-inequalities* for short. They contain as a special case (when there is only one path) the KC-inequalities of [CS94]. We describe the path-bridge inequalities without invoking the notion of required edges. Let $P \geq 0$ and $n_p \geq 2$, $p = 1, \dots, P$, be integers and let there be a partition of the node set of G into sets B^j_p, $j = 1, \dots, n_p$, $p = 1, \dots, P$, and A, Z, where the last two sets are allowed to be empty. The following conditions must hold: *path-bridge, PB; KC*

1. Each of the B^p_j, $j = 1, \dots, n_p$, $p = 1, \dots, P$, is a non-empty union of R-sets.
2. For the parities we require $P + \mathbf{t}(A) = 1 \bmod 2$. If A is a union of R-sets (which implies that so is Z), then we require $P \geq 3$.

For ease of notation we let $B^p_0 := A$ and $B^p_{n_p+1} := Z$ for all $p = 1, \dots, P$; but note that these sets can be empty. We say that $B^p_1, \dots, B^p_{n_p}$ is the p-th *path* of the configuration. See Fig. 2.1(a) for an illustration. The coefficients on the edges are defined as follows. *path*

$$c_e := \begin{cases} 1, & \text{if } e \in (A : Z) \\ \frac{|l-j|}{n_p-1}, & \text{if } e \in (B_j^p : B_l^p), \text{ for } (j,l) \neq (0,n_p),(n_p,0) \\ \frac{1}{n_p-1} + \frac{1}{n_q-1} + \left| \frac{j-1}{n_p-1} - \frac{l-1}{n_q-1} \right|, & \text{if } e \in (B_j^p : B_l^q), \text{ for } p \neq q \\ & \quad\quad \text{and } (j,l) \neq (0,n_q),(n_p,0) \\ 0, & \text{if } e \in E(B_j^p), \text{ for all } j,p. \end{cases} \tag{2.2}$$

The right hand side of the inequality is $\gamma := 1+\sum_{p=1}^{P} \frac{n_p+1}{n_p-1}$. Under these conditions, the inequality (c,γ) is facet-defining for $\mathrm{GRP}(\varGamma,\infty)$, if each of the induced subgraphs $G[B_j^p]$, $p = 1,\dots,P$, $j = 0,\dots,n_p+1$, are non-empty and connected [Let96, CS94]. If the sets A and Z are empty, then some additional conditions are required, in particular $P \geq 2$, see also [CFN85].

We note that, since we allow $P = 0$, the class of PB-inequalities includes the class of R-odd cut inequalities. A path-bridge inequality is called *n-regular* (or simply *regular*) if $n_p = n$ for $p = 1,\dots,P$. It is called *simple*, if $\left|B_j^p\right| = 1$ for all $j = 1,\dots,n_p$, $p = 1,\dots,P$.

regular

simple

KC-inequality

As mentioned above, if $P = 1$, we obtain the class of *KC-inequalities.* We then omit the upper index, write $K := n+1$, and denote the sets by $B_0,\dots,B_K$. KCs are only valid if $B_0 \cup B_K \neq \emptyset$.

Honeycomb (HC-) inequalities

Honeycomb inequalities were first discussed in [CS98]. We briefly review the definition, but we avoid the notion of required edges and instead will only speak of the R-set partition and parity function. The following terminology will be generally helpful.

2.1.9 Definition Let $\mathcal{D}$ be an arbitrary partition of a set V, and let $\mathcal{U}$ be a set of subsets of V. We define the $\mathcal{D}$*-graph* of $\mathcal{U}$ as follows: its node set is $\mathcal{U}$, and two $U \sim U'$ holds iff there exists a $D \in \mathcal{D}$ with $U \cap D, U' \cap D \neq \emptyset$ or $U \cap U' \neq \emptyset$. We say that the set $\mathcal{U}$ is $\mathcal{D}$*-connected,* if the $\mathcal{D}$-graph is connected.

$\mathcal{D}$-connected

Let numbers $K > L > 0$ and $n_k \geq 2$, $k = 1,\dots,K$ be given and a partition of $V(G)$ into sets

$$\begin{aligned} &B_r^k \;\text{ for } r = 1,\dots,n_k \text{ and } k = 1,\dots,L, \\ &B_1^k \;\text{ for } k = L+1,\dots,K. \end{aligned}$$

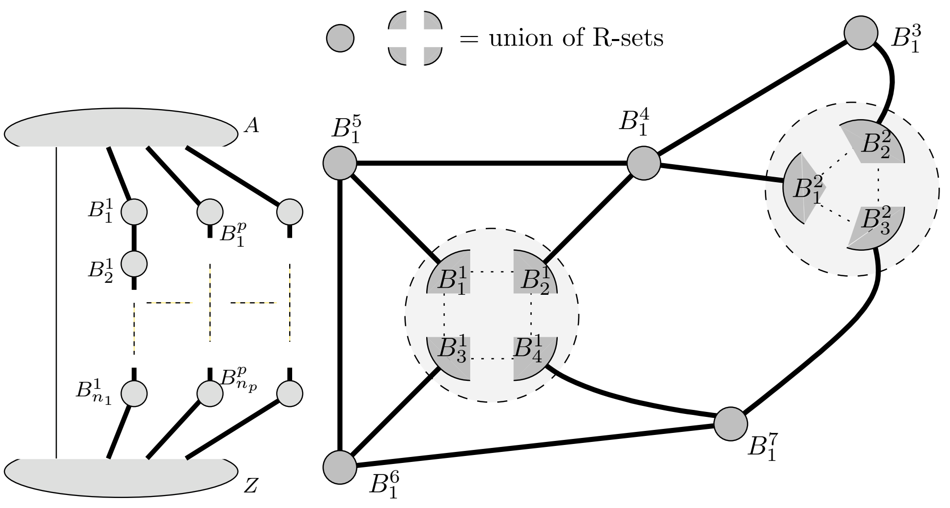
We construct a simple graph G_B by shrinking in G each of the sets B_r^k into one node which we will again denote by B_r^k. Thus G_B has $K - L + n_1 + \dots + n_L$ nodes. Now let T be a spanning tree in G_B. See Fig. 2.1(b) for an illustration. The following conditions are expected to hold:

1. The leaves of the tree $\mathcal{T}$ are precisely the nodes B_r^k, $r = 1,\dots,n_k$, $k = 1,\dots,L$.
2. For any $k = 1,\dots,L$, the distance $\mathrm{dist}_{\mathcal{T}}(B_{r_1}^k, B_{r_2}^k)$ between any two distinct nodes $B_{r_1}^k$ and $B_{r_2}^k$ in the tree $\mathcal{T}$ is greater than or equal to three.
3. Each of the sets $\bigcup_{r=1}^{n_k} B_r^k$, for $k = 1,\dots,L$, and B_1^k, $k = L+1,\dots,K$, is a union of R-sets.
4. The sets B_r^k are even, i.e., $\mathbf{t}(B_r^k) = 0$.
5. For all $k \in \{1,\dots,L\}$ the set $\{B_i^k \mid i = 1,\dots,n_k\}$ of node sets is $\mathcal{C}$-connected.

The condition in [CS98] which uses required edges is replaced by 4 and 5. A node partition and tree of this kind is called a *honeycomb* configuration. The coefficients are defined as follows:

honeycomb, HC

$$c_e := \begin{cases} \mathrm{dist}_{\mathcal{T}}(B_{r_1}^k, B_{r_2}^k) - 2, & \text{if } e \in (B_{r_1}^k : B_{r_2}^k), \text{ with } r_1 \neq r_2 \\ \mathrm{dist}_{\mathcal{T}}(B_{r_1}^{k_1}, B_{r_2}^{k_2}), & \text{if } e \in (B_{r_1}^{k_1} : B_{r_2}^{k_2}), \text{ with } k_1 \neq k_2 \\ 0, & \text{otherwise, i.e., if } e \in E(B_r^k). \end{cases}$$



(a) *Path-bridge configuration.* (b) *Honeycomb configuration with $L = 2$ and $K = 7$.*

Figure 2.1: Definition of honeycomb and path-bridge configurations.

Corberán and Sanchis [CS98] proved the validity and facet-defining property for $\mathrm{GRP}(\Gamma, \infty)$ of the honeycomb inequality $(c, 2(K-1))$. We note that the definition in [CS98] is more general: the condition 1 is weakened, but sequential lifting is required to obtain the coefficients on some edges. We have included the condition 1 because it greatly reduces the technicality of the definition.

Facets from the Graphical Traveling Salesman Polyhedron

Invoking 0-node lifting 2.1.1, it is easy to see the truth of the following proposition.

2.1.10 Proposition ([CS98]) *Any facet-defining inequality for GTSP($G^{\mathrm{s}}_{\mathcal{C}}$) can be made into a facet-defining inequality of* $\mathrm{GRP}(\Gamma, \infty)$ *applying 0-node lifting in the following way: every node C of $G^{\mathrm{s}}_{\mathcal{C}}$ is replaced by the set of nodes C of G.*

2.2 The Ghiani-Laporte tree and bounded polyhedra

Let c be a vector of edge costs for G, and denote by T the edge set of any minimum spanning tree of $G^{\mathrm{m}}_{\mathcal{C}}$, with respect to the cost vector c restricted to the R-external edges. Ghiani & Laporte [GL00] showed that $\min\{cx \mid x \in \mathscr{S}(\Gamma, \infty)\} = \min\{cx \mid x \in \mathscr{S}(\Gamma, \infty), x \leq \mathbf{b}^T\}$, where $\mathbf{b}^T := \mathbf{1} + \chi^T$ is a vector of upper bounds. This means that, without changing the optimum solution value, we can bound the number of times an edge is contained in a semitour by 1, if the edge is not in the tree T, and by 2 if it is.

2.2.1 Theorem *Let Γ be a GRP-structure b a bounds vector and c a vector of edge costs. If $\{e \in E(G) \mid b_e \geq 2\}$ contains the edge set of a c-minimum spanning tree of $G^{\mathrm{m}}_{\mathcal{C}}$, then optimizing over* $\mathrm{GRP}(\Gamma, b)$ *gives an optimal solution to the General Routing Problem instance given by Γ and c.* □

2.2.2 Definition If $\Gamma = (G, \mathcal{C}, \mathbf{t})$ is a GRP-structure and T is the edge set of a spanning tree of $G^{\mathrm{m}}_{\mathcal{C}}$, we say that the quadruple $(G, \mathcal{C}, \mathbf{t}, T)$ is a *Ghiani-Laporte GRP-structure* or *GL-GRP-structure* for short. If a specific GRP instance is under consideration and T is a minimum spanning tree as just described, we say that T is a *Ghiani-Laporte tree.* *Ghiani-Laporte tree, GL-GRP-structure*

2.2.3 Remark When optimizing a non-negative cost function over a relaxation of one of the bounded polyhedra $\mathrm{GRP}(\mathit{\Gamma}, b)$, only inequalities which are valid and facet-defining for $\mathrm{GRP}(\mathit{\Gamma}, \infty)$ are needed to achieve the optimal lower bound.

2.2.1 The Ghiani-Laporte polytope

Let T be a Ghiani-Laporte tree. Denote by $G^T := G + T$ the loopless multigraph which results if, for every edge $e \in T$, we add a "duplicate" edge $e^\sim$ with the same end nodes. Hence $V(G^T) = V(G)$ and

$$E(G^T) = E(G) \cup \{e^\sim \mid e \in T\}.$$

If the duplicates $e^\sim$ are assigned the same cost as e, i.e., $c_{e^\sim} := c_e$, then, by considering semitours on G^T instead of on G, we can restrict our attention to semitours x with $x \le \mathbf{1}$. The convex hull of all semitours x on G^T with $x \le \mathbf{1}$ is a 0/1-polytope which was introduced in [GL00]. Therefore we call it the *Ghiani-Laporte polytope* and denote it by $\mathrm{GL}(\mathit{\Gamma}, T)$. Note that if we let $\tilde{\mathit{\Gamma}} := (G^T, \mathcal{C}, \mathbf{t})$, then $\mathrm{GL}(\mathit{\Gamma}, T) = \mathrm{GRP}(\tilde{\mathit{\Gamma}}, \mathbf{1})$. [GL00] gave an IP-formulation for optimizing over $\mathrm{GL}(\mathit{\Gamma}, T)$. Besides the non-negativity inequalities $x \ge \mathbf{0}$ and the upper bounds $x \le \mathbf{1}$, it consists of the connectivity inequalities (1.1b) and the so-called *cocircuit inequalities*:

Ghiani-Laporte polytope $\mathrm{GL}(\mathit{\Gamma}, T)$

cocircuit inequality

$$x(\partial_{G^T}(U) \setminus F) - x(F) \ge 1 - |F| \tag{2.3}$$
$$\text{for } U \subsetneq V(G),\ U \ne \emptyset \text{ and } F \subseteq \partial_{G^T}(U) \text{ with } |F| + \mathbf{t}(U) \text{ odd.}$$

These inequalities are valid for $\mathrm{GL}(\mathit{\Gamma}, T)$ [GL00]. Strictly speaking, we should require that $\partial_{G^T}(U)$ really be a cocircuit. For the IP-formulation, only the cocircuit inequalities with $|U| = 1$ are needed.

In [GL00] this IP-formulation was used in a Branch-and-Cut algorithm which produced promising computational results. In that paper, an attempt was made to study the the polytope $\mathrm{GL}(\mathit{\Gamma}, T)$, too. However, little is known about its theoretical properties. Thus, while the polytope $\mathrm{GL}(\mathit{\Gamma}, T)$ has proven very useful in practice, little is known about its theoretical properties, or any of the polyhedra $\mathrm{GRP}(\mathit{\Gamma}, b)$. These polyhedra are at the focus of interest of the following two chapters.

Note that the polytope $\mathrm{GL}(\mathit{\Gamma}, T)$ has a redundant symmetry: for each $e \in T$ if we take a point $x \in \mathrm{GL}(\mathit{\Gamma}, T)$ and exchange the values of x_e and $x_{e^\sim}$, we get "essentially" the same point. It turns out that this extra symmetry of the polytope makes its study more difficult than necessary. However, it turns out that this difficulty can be avoided by examining the polytope $\mathrm{GRP}(\mathit{\Gamma}, \mathbf{b}^T)$, and then transferring the results back to $\mathrm{GL}(\mathit{\Gamma}, T)$.

2.2.2 Configuration inequalities

Now we contribute some basic facts about the general polyhedra $\mathrm{GRP}(\mathit{\Gamma}, b)$, where $\mathit{\Gamma} = (G, \mathcal{C}, \mathbf{t})$ and $b\colon E(G) \to \mathbb{Z}_+^* \cup \{\infty\}$. We start by showing that for the polyhedra with bounds, Corollary 2.1.4 need not hold: Not every facet-defining inequality for $\mathrm{GRP}(\mathit{\Gamma}, b)$ is a configuration inequality. Fig. 2.2 shows the coefficients of an inequality which defines a facet, in which two parallel edges have distinct (and even non-zero) coefficient. The right hand side is -4. The graph is K_5 plus one edge, all R-sets have cardinality one, and all nodes are even. In order for the inequality to define a facet, some of the edges with coefficient two or minus two must have upper bound two, and the remaining edges must have upper bound one.

We come back to 0-node lifting for the case of finite bounds. An alternative definition to the one on page 16 would be to shrink G° to a loopless multigraph $G := G^\circ/F$, and define

$$b := b^\circ|_{E(G)}\ , \text{ and } a^\circ \text{ such that } \ a^\circ|_{E(G)} = a,\ a^\circ|_F = 0. \tag{2.4}$$

At present, if $b < \infty$, we do not have the tools at our hands to understand the relationship between the two concepts, or the facet-defining property of a°. These issues will be addressed in Chapter 4.

Lemma 2.1.6 has a counterpart for polyhedra with bounds, whose prove can be accomplished with the elementary methods which we have at our hands in this chapter. (We will develop

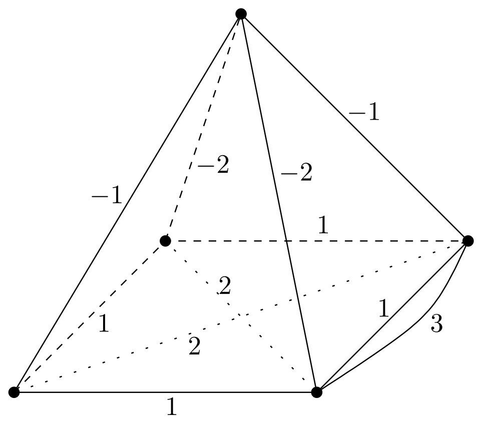


Figure 2.2: Non-configuration inequality defining a facet of $\mathrm{GRP}(\Gamma, b)$.

more subtle tools for questions of this kind in Chapter 4.) Let (a, α) be a valid configuration inequality for $\mathrm{GRP}(\Gamma, b)$, and denote by $\Gamma^\bullet = (G^\bullet, \mathcal{C}^\bullet, \mathbf{t}^\bullet)$ the GRP-structure which results from contracting each edge in e with $a_e = 0$, where the contraction is in the class of loopless multigraphs. Further, let $a^\bullet := a_{|E(G^\bullet)}$, and $b := b_{|E(G^\bullet)}$. For the case that the polyhedron $\mathrm{GRP}(\Gamma, b)$ is full-dimensional, since contraction preserves this property (see Chapter 4), the following lemma shows that configuration inequalities are 0-node lifted from smaller GRP-structures.

2.2.4 Lemma *If (a, α) is a configuration inequality, then every inequality which dominates $(a^\bullet, \alpha)$, when 0-node lifted, dominates (a, α) .* □

2.3 Results on polyhedra for few R-sets

In this section, we summarize results on the bounded and unbounded T-join polyhedra, which are identical to $\mathrm{GRP}(\Gamma, b)$ if there is only one R-set, i.e., $\mathcal{C} = \{V\}$. Then we give results on the case where there are two R-sets. For the following proposition, see, e.g., [Sch03].

2.3.1 Proposition ([Edm65, EJ77, Pul73]) *If there is only one R-set, then we have the following complete description of* $\mathrm{GRP}(\Gamma, b)$*:*

$$x(\partial(U) \setminus F) - x(F) \geq 1 - b(F) \tag{2.5a}$$
$$\text{where } \emptyset \neq U \subsetneq V(G) \text{ and } F \subseteq \partial(U) \cap E(G) \text{ with } \mathbf{t}(U) + b(F) \text{ odd}$$
$$b \geq x \geq 0 \tag{2.5b}$$

The inequalities (2.5a) are called *blossom inequalities,* and they are of course valid for $\mathrm{GRP}(\Gamma, b)$ without restrictions on the R-set partition. We note that they contain the *cut upper-bound constraints* $x(\partial(U)) \leq b(U) - 1$ if U is a union of R-sets and $b(\partial(U))$ odd, which by were introduced to the GRP by Letchford [Let03].

blossom inequalities

2.3.1 Some new facets

Since the one R-set polyhedra are fairly simple in terms of classes of facets, it is believed or hoped that this might be true for the 2-R-set polyhedra, too. A conjecture [Let05a] inspired by [CS94, San90] says that non-negativity, R-odd cut, and the unique connectivity inequality might form a complete description of the polyhedron. This is not at all the case: the polyhedra for GRP-structures with two R-sets can, in general, be very complicated.

We contribute to the area of "polyhedral botany" for the General Routing Problem, i.e., we exhibit some families of new, strange, and apparently useless facet-defining inequalities. We give

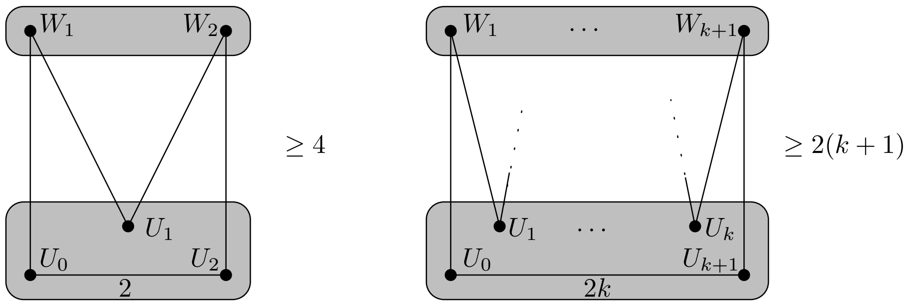


Figure 2.3: Facets of $\mathrm{GRP}(\Gamma,\infty)$ with two R-sets: M-configurations.

two general families and three individuals of facet-defining inequalities for $\mathrm{GRP}(\Gamma,\infty)$ with $|\mathcal{C}| = 2$. There is no reason to believe that there is only a small variety of facets of this polyhedron.

Given that the General Routing Problem is solvable in polynomial time if $|\mathcal{C}| = 2$ (see 1.2), it is surprising that the polyhedra have so many apparently completely different facets. It might be intriguing to search for a complete set of facets for these polyhedra.

Let a configuration of the following form be given. For an integer $k \geq 1$, suppose that the node set of G is partitioned into sets $U_0, \ldots, U_{k+1}, W_1, \ldots, W_{k+1}$, such that the two sets $\bigcup U_j$ and $\bigcup W_j$ are both unions of R-sets, and

$$\begin{aligned} \mathbf{t}(U_0) = \mathbf{t}(U_{k+1}) &= 1 \mod 2 & \\ \mathbf{t}(U_j) \quad &= 0 \mod 2 \quad & \text{for } j = 1, \ldots, k \\ \mathbf{t}(W_j) \quad &= 0 \mod 2 \quad & \text{for } j = 1, \ldots, k+1. \end{aligned} \tag{2.6}$$

Suppose that the set $\{U_0, \ldots, U_{k+1}\}$ of subsets of $V(G)$ is $\mathcal{C}$-connected (see Def. 2.1.9) and the same holds for the set $\{W_1, \ldots, W_{k+1}\}$. See Fig. 2.3 for an illustration. We define the coefficients by

$$a_e = \begin{cases} 0 & \text{if } e \in E(U_j) \text{ or } e \in E(W_j) \\ 1 & \text{if } e \in (U_j : W_{j+1}) \cup (W_j : U_j) \\ 2k & \text{if } e \in (U_0 : U_{k+1}); \end{cases}$$

the coefficients of the remaining edges are the lengths of the shortest paths with respect to these coefficients (cf. Def. 2.1.1). We call such a configuration an *M-configuration* and the inequality $(a, 2(k+1))$ an *M-inequality.* We have the following fact, which is proved by standard arguments which we omit. *M-inequality*

2.3.2 Proposition *M-inequalities are valid for* $\mathrm{GRP}(\Gamma,\infty)$. *If the induced subgraphs* $G(U_j)$ *and* $G(W_j)$ *are connected, then they are facet-defining for* $\mathrm{GRP}(\Gamma,\infty)$,

Now we define another new family of facet-defining inequalities for $\mathrm{GRP}(\Gamma,\infty)$. See Fig. 2.4 for an illustration. Let $k \geq 1$ be an integer, and let $U_0, \ldots, U_{k+1}$, $W_1, \ldots, W_{k+1}$, V be a the sets of a partition of $V(G)$. Suppose that both sets $V \cup \bigcup U_j$ and $\bigcup W_j$ are unions of R-sets, that V is an R-even set and (2.6) holds. Further, suppose that the two sets $\{V, U_0, \ldots, U_{k+1}\}$ and $\{W_1, \ldots, W_{k+1}\}$ of subsets of $V(G)$ are both $\mathcal{C}$-connected. We define the coefficients of an inequality by

$$a_e = \begin{cases} 0 & \text{if } e \in E(V) \text{ or } e \in E(U_j) \text{ or } e \in E(W_j) \\ 1 & \text{if } e \in (U_j : W_{j+1}) \cup (W_j : U_j) \\ k & \text{if } e \in (U_0 : V) \text{ or } e \in (V : U_{k+1}) \\ k + 2 - 2\min(l, k+1-l) & \text{if } e \in (V : U_j) \text{ with } j \in \{1, \ldots, k\}. \end{cases}$$

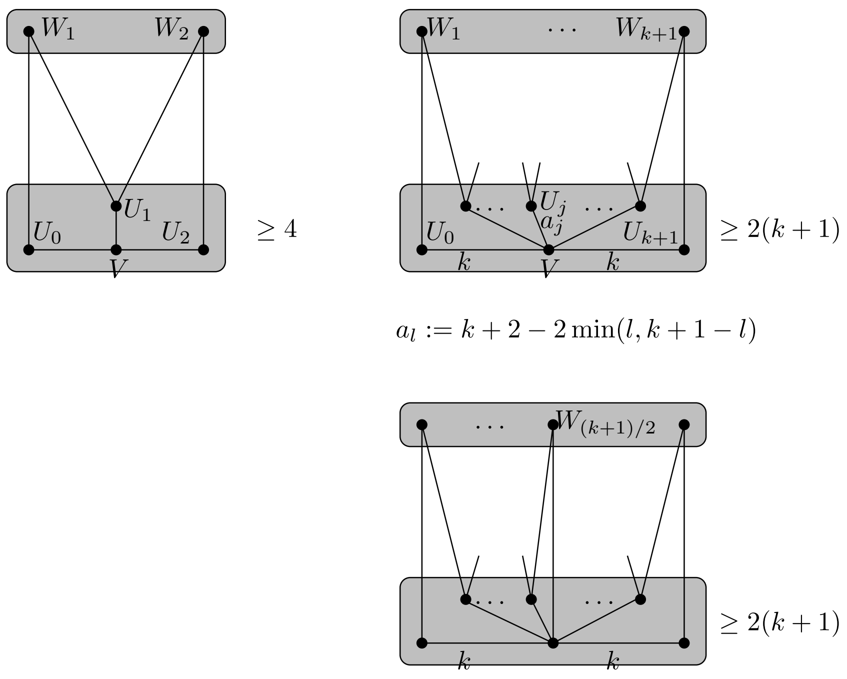


Figure 2.4: Facets of $\mathrm{GRP}(\Gamma,\infty)$ with two R-sets: M-bounce inequalities.

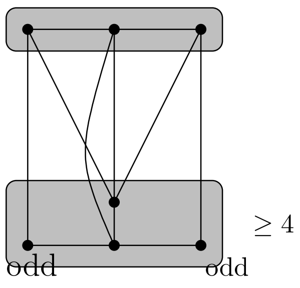


Figure 2.5: Facets of $\mathrm{GRP}(\Gamma,\infty)$ with two R-sets: M-top inequalities.

If k is even, then the coefficients of the remaining edges are the lengths of the shortest-path with respect to the coefficients of these edges. If k is odd, then

$$a_e = 1 \quad \text{if } e \in (V : W_{(k+1)/2}),$$

and the then remaining edges get shortest paths lengths. We call such a configuration an *M-bounce configuration,* and we call the inequality $(a, 2(k+1))$ an *M-bounce inequality.* We have the following result, again we spare the reader the proof.

M-bounce inequality

2.3.3 Proposition *M-bounce inequalities are valid for* $\mathrm{GRP}(\Gamma,\infty)$*, and they are facet-defining if the induced subgraphs $G(V)$, $G(U_j)$, and $G(W_j)$ are connected.*

We close this section with some more examples of facet defining inequalities for the two-R-set case, which we display in Figures 2.5 and 2.6. In both figures, the edges which are drawn have coefficient 1 in the corresponding inequality, and the edges which are not drawn have shortest-paths lengths. As in the M- and M-bounce families, the *M-top*-inequality of Fig. 2.5 has only two R-odd sets, while the *N1-* and *N2*-inequalities of Fig. 2.6 have four R-odd sets.

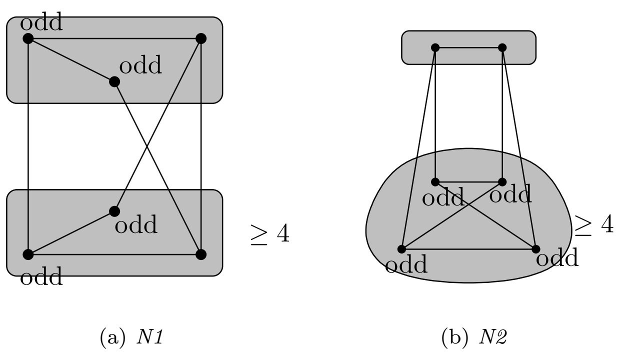


Figure 2.6: Facets of GRP($\mathit{\Gamma},\infty$) with two R-sets: N-inequalities.

2.4 An even smaller polytope

In this section, we show how the set of feasible solutions can be reduced to a number of 2^{m-n+1}, where $n := |V(G)|$ and $m := |E(G)|$. Though theoretically, this polyhedron appears to be be even more complex than GRP($\mathit{\Gamma}, \mathbf{1}$), it has the advantage that the number of feasible solutions is known and they can be easily enumerated in $O(m2^{m-n+1})$ time. The idea is based on the following extension of Theorem 2.2.1.

2.4.1 Proposition *Let x be an optimal solution to the GRP-instance defined on $\mathit{\Gamma}$ with costs $c \geq 0$ which is minimal under the condition $x \in \mathscr{S}(\mathit{\Gamma},\infty)$ with respect to component-wise vector comparison. Then*

(a). $x \in \{0,1,2\}^{E(G)}$,

(b). the edges with $x_e = 1$ form a T-join in G, with T the set of odd nodes,

(c). the edges with $x_e = 2$ are R-external and bridges in $G(x)$, and

(d). if all R-internal edges and all edges e with $x_e = 1$ are contracted, then the edges with $x_e = 2$ form a c-minimum spanning tree in the resulting graph. □

Let us assume, for the sake of the simplicity of the formulation, that there are no two edges with the same cost. Then, for every T-join x', with T the set of R-odd nodes, there exists a unique set of edges E_2 such that $x' + 2\chi^{E_2}$ satisfies (a-d). On the other hand, every minimal optimal solution can be decomposed into x' and χ^{E_2}, as the lemma shows. Since the number of T-joins of a connected graph is 2^{m-n+1}, we obtain the same number of semitours.

Chapter 3

Transformation and symmetry

3.1 Symmetry and isomorphism of GRP polyhedra

We now introduce a transformation method which allows one to create new facets from known ones. Let E be a set and $b\colon E \to \mathbb{Z}_+^* \cup \{\infty\}$ be a bound vector. For $y \in \{0,1\}^E$ with $y_e = 0$ for all e with $b_e = \infty$ we define the following mapping:

$\mathrm{flip}^{[b;y]}$

$$\mathrm{flip}^{[b;y]}\colon \mathbb{R}^E \to \mathbb{R}^E \colon x \mapsto y \odot b + x^{[y]}$$

where, for all $e \in E$,

$x \odot y, x^{[y]}$

$$\big(y \odot b\big)_e := y_e b_e, \qquad \text{and} \qquad \big(x^{[y]}\big)_e := \begin{cases} x_e & \text{if } y_e = 0 \\ -x_e & \text{if } y_e = 1, \end{cases}$$

or, in other words, if $y_e = 0$ then $(\mathrm{flip}^{[b;y]}(x))_e = x_e$, and if $y_e = 1$ then $(\mathrm{flip}^{[b;y]}(x))_e = b_e - x_e$.

3.1.1 Remark The mapping $\mathrm{flip}^{[b;y]}$ is an affine isomorphism. For two vectors $y_1, y_2 \in \{0,1\}^E$, if $\oplus$ denotes addition modulo two, $\mathrm{flip}^{[b;y_1 \oplus y_2]}$ is equal to the composition of the mappings $\mathrm{flip}^{[b;y_1]}$ and $\mathrm{flip}^{[b;y_2]}$, while $\mathrm{flip}^{[b;\mathbf{0}]}$ is the identity mapping.

Let $\Gamma = (G, \mathcal{C}, \mathbf{t})$ be a GRP-structure, b a bound vector, and $y \in \{0,1\}^{E(G)}$ such that

$$y_e = 0 \qquad \text{whenever } e \notin E_{\mathrm{int}}(G) \text{ or } b_e = \infty. \tag{3.1}$$

Define

$$\mathbf{t}_y\colon v \mapsto \mathbf{t}(v) \oplus (y \odot b)(\partial(v)) = \mathbf{t}(v) \oplus \bigoplus_{\substack{e \in \partial(v) \\ b_e \text{ odd}}} y_e$$

$\mathbf{t}_y$

and $\Gamma_y := (G, \mathcal{C}, \mathbf{t}_y)$. Then Γ_y is a GRP-structure and we have

Γ_y

$$x \in \mathscr{S}(\Gamma, b) \text{ if and only if } \mathrm{flip}^{[b;y]}(x) \in \mathscr{S}(\Gamma_y, b) \tag{3.2}$$

Note that $\Gamma_{\mathbf{0}} = \Gamma$ and $\Gamma_{y_1 \oplus y_2} = (\Gamma_{y_1})_{y_2}$. A consequence is the following proposition.

3.1.2 Proposition *Let y and Γ_y be as just described. The mapping $\mathrm{flip}^{[b;y]}$ is an isomorphism between the polyhedra $\mathrm{GRP}(\Gamma, b)$ and $\mathrm{GRP}(\Gamma_y, b)$.*

Proof. Since $\mathrm{flip}^{[b;y]}$ is an affine isomorphism, the statement follows from the fact that, by (3.2), it maps $\mathscr{S}(\Gamma, b)$ onto $\mathscr{S}(\Gamma_y, b)$. □

We can apply the proposition in two ways. If

$$(y \odot b)(\partial(v)) \ = 0 \qquad \text{for every } v \in V(G), \tag{3.3}$$

then the parities remain unchanged and we can study symmetries of the polyhedron $\mathrm{GRP}(\varGamma, b)$.

3.1.3 Corollary *Let m^0_{int} denote the number of R-internal edges e with b_e even, and let H denote the subgraph induced by the R-internal edges e with b_e odd. Let $m^1_{\text{int}} := |E(H)|$, and denote by c^1 the number of connected components of H. The symmetry group of* $\mathrm{GRP}(\varGamma, b)$ *includes a subgroup isomorphic to $\{0,1\}^{m^0_{\text{int}}+m^1_{\text{int}}-c^1}$.*

Proof. The subgroup of $\{0,1\}^{E(G)}$ consisting of all $y \in \{0,1\}^{E(G)}$ which satisfy (3.1) and (3.3) is the cycle space of H. The dimension of this space is $m^1_{\text{int}} - c^1$. □

If we drop condition (3.3), then we have isomorphisms between polyhedra arising from different parity functions.

3.1.4 Corollary *Let G be a graph and $\mathcal{C}$ be a partition of its node set. Further, let b be a bound vector satisfying the conditions that, for every R-set $C \in \mathcal{C}$, the set of edges $e \in E(C)$ with b_e odd spans C. Then all polyhedra* $\mathrm{GRP}(\varGamma, b)$*, with $\varGamma = (G, \mathcal{C}, \mathbf{t})$ for an arbitrary parity function $\mathbf{t}$, are isomorphic.* □

In terms of required edges, this last corollary states that if two sets of required edges define the same R-sets, then their polyhedra are isomorphic (if, for example, all R-internal edges have upper bound one).

3.1.1 Transformation of valid inequalities

switched inequality

The practical value of Proposition 3.1.2 is the following. Given an inequality (a, α) and a vector y as above, consider the inequality $(a^{[y]}, \alpha - (b \odot y)a)$. We say that the inequality (a, α) is *switched* to obtain the inequality $(a^{[y]}, \alpha - (b \odot y)a)$. The slacks of the two inequalities are related by the isomorphism $\mathrm{flip}^{[b;y]}$. For any $x \in \mathbb{R}^{E(G)}$ we have

$$\alpha - (b \odot y)a \ - \ a^{[y]} \, \mathrm{flip}^{[b;y]}(x) = \alpha - ax. \tag{3.4}$$

With this relation, we obtain the following proposition as an immediate consequence of Proposition 3.1.2.

3.1.5 Proposition *We abbreviate $\alpha' := \alpha - (b \odot y)a$.*

(a). The inequality (a, α) is valid for $\mathrm{GRP}(\varGamma, b)$ *if and only if $(a^{[y]}, \alpha')$ is valid for* $\mathrm{GRP}(\varGamma_y, b)$.

(b). The mapping $\mathrm{flip}^{[b;y]}$ *is an isomorphism between the face of* $\mathrm{GRP}(\varGamma, b)$ *induced by (a, α) and the face of* $\mathrm{GRP}(\varGamma_y, b)$ *induced by $(a^{[y]}, \alpha')$.*

In particular, the inequality (a, α) is facet-defining for $\mathrm{GRP}(\varGamma, b)$*, if and only if $(a^{[y]}, \alpha')$ is facet-defining for* $\mathrm{GRP}(\varGamma_y, b)$. □

If y ranges over all elements of $\{0,1\}^{E(G)}$ which satisfy (3.1) and (3.3) then the switched inequalities $(a^{[y]}, \alpha - (b \odot y)a)$ form a symmetry class of the original inequality (a, α).

If, more generally, (b, β) is any inequality, and there exists a y with (3.1) and an inequality (a, α) which is valid (facet-defining) for $\mathrm{GRP}(\varGamma_y, b)$ such that $b = a^{[y]}$ and $\beta = \alpha - (b \odot y)a$, then we know that (b, β) is valid (facet-defining) for $\mathrm{GRP}(\varGamma, b)$. In particular, when examining the polyhedron and facets, we can assume without loss of generality that all nodes have even parity, i.e., $\mathbf{t}(u) = 0$ for all $u \in V(G)$.

3.2 Relaxation of valid inequalities

We start with a definition which will be useful in Chapter 6, too.

3.2.1 Definition Let $\mathcal{U}$ be a partition of $V(G)$, and let $f = vw \in E(G)$. If U_v and U_w denote the sets which contain v and w respectively, we define a partition $\mathcal{U} \circledcirc f$ by replacing in $\mathcal{U}$ the sets U_v and U_w by their union $U_v \cup U_w$. In other words, *$\mathcal{U} \circledcirc f$, $\mathcal{U} \circledcirc F$*

$$\mathcal{U} \circledcirc f := (\mathcal{U} \setminus \{U_v, U_w\}) \cup \{U_v \cup U_w\}.$$

For a set $F = \{f_1, \dots, f_k\} \subseteq E(G)$, we define $\mathcal{U} \circledcirc F := \mathcal{U} \circledcirc f_1 \circledcirc \dots \circledcirc f_k$.

This section is based on the following observation.

3.2.2 Remark We have $\mathscr{S}(\mathit{\Gamma}, b) \subseteq \mathscr{S}(G, \mathcal{C} \circledcirc F, \mathbf{t}, b)$. Hence the transition from $\mathcal{C}$ to $\mathcal{C} \circledcirc F$ is a relaxation.

Now, if $b_f < \infty$ for all $f \in F$, abbreviate $y := \chi^F \in \{0,1\}^{E(G)}$ and define $\mathit{\Gamma}_F := (G, \mathcal{C} \circledcirc F, \mathbf{t}_y)$. Suppose that (a, α) is a valid inequality for GRP$(\mathit{\Gamma}_F, b)$. Then it follows from Proposition 3.1.5 that the inequality $(a^{[y]}, \alpha - (b \odot y)a)$ is valid for GRP$(G, \mathcal{C} \circledcirc F, \mathbf{t}, b)$. By the remark above, it is also valid for GRP$(\mathit{\Gamma}, b)$.

We still say that the derived inequality $(a^{[y]}, \alpha - (b \odot y)a)$ is obtained by *switching* the valid inequality (a, α). The switching method of Section 3.1 clearly is the special case when $F \subseteq E_{\text{int}}$. Only in this case do we get the facet-defining property of $(a^{[y]}\alpha - (b \odot y)a)$ for free if (a, α) is facet-defining. If F contains an R-external edge, it is not guaranteed that the switched inequality defines a face of high dimension. However, we note that there are cases when the classical form of the so-called Path-Bridge inequalities, which we will define in Section 3.3.2, is dominated by a switched variant [Let03]. Moreover, in [RT06], we show that switched Path-Bridge inequalities even dominate connectivity inequalities (1.1b) in certain situations. We will address the issue of facet-defining property of switched forms of some classes of valid inequalities in the next section. *switched inequality*

3.2.3 Remark Using the notion of required edges and assuming that the edges in F are not required, the switching process can be described as follows:

1. Make the edges in F required.
2. Find an inequality (a, α) which is valid in this new situation.
3. Switch the inequality (a, α) according the Section 3.1. The resulting inequality is valid for the polyhedron defined with the original set of required edges.

The analogy fails if $F \cap E_R \neq \emptyset$.

All the classical inequalities discussed in this paper have in common that they require a GRP-structure with at least some non-trivial R-sets, i.e., R-sets C with $|C| \geq 2$. However, if the set F is chosen appropriately, the resulting inequalities are valid for the 0/1-polytope of the Graphical TSP, i.e., the polytope which is defined as the convex hull of all incidence vectors of spanning (connected) Eulerian subgraphs of the graph G. Thus our results have consequences for polytopes associated with TSP.

3.3 Examples

In this section we give examples. We start with two trivial ones. First, the bound inequalities are related by switching: $x_e \leq b_e$ can be obtained by switching the inequality $x_e \geq 0$. Second, all blossom inequalities (see (2.5a) on page 21) are switched R-odd cut inequalities.

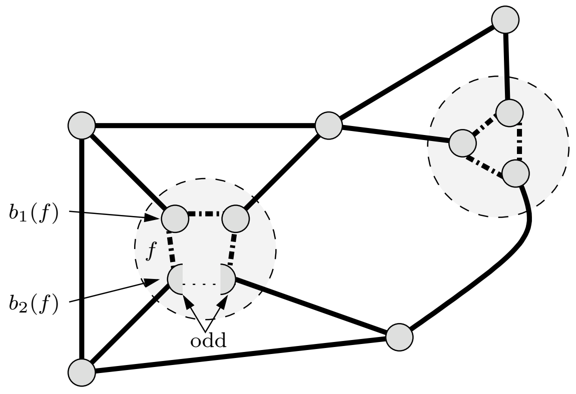


Figure 3.1: Switched honeycomb

3.3.1 Switched honeycombs

switched HCs For the definition of the *switched honeycomb (HC-) inequalities,* in addition to the partition of the node set into sets B_r^k, as defined in 2.1.2 on page 18, we need sets of edges F^k, $k = 1, \dots, K$, such that $b_f < \infty$ for all $f \in F^k$, $k = 1, \dots, L$. Further, we require that for each $f \in F^k$ there exist two distinct numbers $r_1(f), r_2(f) \in \{1, \dots, n_k\}$ with $f \in (B^k_{r_1(f)} : B^k_{r_2(f)})$. See Fig. 3.1. We replace items 4 and 5 of the definition on page 18 by

4a. The relation $\mathbf{t}(B_r^k) + b(F^k \cap \partial(B_r^k)) = 0 \bmod 2$ holds for each $r = 1, \dots, n_k$ and $k = 1, \dots, L$.

5a. For all $k \in \{1, \dots, L\}$ the set $\{B_i^k \mid i = 1, \dots, n_k\}$ of node sets is $\mathcal{C} \circledcirc F^k$-connected.

We give the coefficients of the switched honeycomb inequality:

$$c_e := \begin{cases} \operatorname{dist}_T(B^k_{r_1}, B^k_{r_2}) - 2 & \text{if } e \in (B^k_{r_1} : B^k_{r_2}) \setminus F^k, \text{ for } r_1 \neq r_2 \\ 2 - \operatorname{dist}_T(B^k_{r_1(f)}, B^k_{r_2(f)}) & \text{if } e \in F^k \\ \operatorname{dist}_T(B^{k_1}_{r_1}, B^{k_2}_{r_2}) & \text{if } e \in (B^{k_1}_{r_1} : B^{k_2}_{r_2}), \text{ for } k_1 \neq k_2 \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

The right hand side is

$$\gamma := 2(K-1) + 2\sum_k b(F^k) - \sum_k \sum_{f \in F^k} b_f \operatorname{dist}_T(B^k_{r_1(f)}, B^k_{r_2(f)}). \tag{3.6}$$

The switched honeycomb inequality (c, γ) is valid for $\mathrm{GRP}(\mathit{\Gamma}, b)$. By Proposition 3.1.5, the inequality defines a face of $\mathrm{GRP}(\mathit{\Gamma}, b)$ which has the same dimension as the face of $\mathrm{GRP}(\mathit{\Gamma}_{\chi^{F \cap E_{\text{int}}}}, b)$ induced by the inequality (c', γ'), where c' and γ' are defined as in (3.5) and (3.6), but with F^k replaced by $F^k \setminus E_{\text{int}}$ for all k. Hence, if $F \subseteq E_{\text{int}}$, then Proposition 3.1.5 shows that the inequality defines a facet of the polyhedron with bounds under similar conditions as are required for the classical honeycomb inequalities to define facets (we will address this topic in 4.4.3). For the case that $F \setminus E_{\text{int}} \neq \emptyset$, it can be shown that the switched honeycomb inequalities define facets of $\mathrm{GRP}(\mathit{\Gamma}, b)$ under the same conditions.

3.3.2 Switched path-bridge inequalities

switched PBs We come to the description of *switched path-bridge inequalities.* Let P, n_p, A, Z, B_j^p be as in 2.1.2, and $F \subseteq (A : Z)$, but with the condition 2 replaced by

2a. $P + \mathbf{t}(A) + b(F) = 1 \bmod 2$, and if A is a union of R-sets, then $P + b(F) \geq 3$ must hold.

Let c be the vector of coefficients as defined in (2.2) on page 18, but with the coefficients on edges $f \in F$ changed from 1 to -1. Then $(c, 1 - b(F) + \sum_{p=1}^{P} \frac{n_p+1}{n_p-1})$ is a switched PB-inequality and hence valid for $\mathrm{GRP}(\mathit{\Gamma}, b)$. If we allow $P = 0$, then the definition includes the blossom inequalities (2.5a) (see page 21) as a subclass. We note that the simplest form of switched PB-inequalities, namely the switched 2-regular PBs with $b = \mathbf{1}$, were found independently by Letchford [Let03].

It can be shown that the switched PB-inequalities define facets under similar conditions which are sufficient for the classical PB-inequalities to define facets of the polyhedron with bounds, see 4.4.2. We will give a stronger result for the facet defining property of switched PB-inequalities based on a more elegant argument. We can assume that the set F does not include R-internal edges, because we can use Proposition 3.1.5 for these edges. Further, for simplicity of the presentation, we restrict the exposition to the case that $b_f = 1$ for all $f \in F$.

The idea of the following theorem is to turn one edge $f \in F$ into a path. It can be used inductively on $|F|$. Let P, n_p, A, Z, B_j^p, and F as just defined and let (c, γ) be the corresponding switched PB-inequality. Let $f \in F$ be an R-external edge.

Construct a graph G^f out of G in the following manner. Denote the end nodes of f by w_0 and w_3, and subdivide the edge f twice by adding new nodes w_1 and w_2.[1] In other words, the edge f is replaced by two new nodes w_1 and w_2 and three edges

$$e_0 = w_0 w_1 \qquad e_1 = w_1 w_2 \qquad e_2 = w_2 w_3.$$

This means that the edge e_i has the end nodes w_i and w_{i+1}, for $i = 0, 1, 2$. See Fig. 3.2 for an illustration. Define a partition $\mathcal{C}^f$ of the node set of G^f by

$$\mathcal{C}^f := \mathcal{C} \cup \{\{w_1\}, \{w_2\}\},$$

and a parity function by letting $\mathbf{t}^f(v) := \mathbf{t}(v)$ for all $v \in V(G)$ and $\mathbf{t}^f(w_i) := 0$ for $i = 1, 2$. Let $\mathit{\Gamma}^f := (G^f, \mathcal{C}^f, \mathbf{t}^f)$, and define bounds by $b^f_e = b_e$ for all $e \in E(G^f) \setminus \{e_0, e_1, e_2\}$ and $b^f_{e_i} = 2$.

We modify the path-bridge configuration on $\mathit{\Gamma}$ to obtain a path-bridge configuration on $\mathit{\Gamma}^f$ by adding an extra path $P + 1$:

$$B_0^{P+1} := A, \;\; B_1^{P+1} := \{w_1\}, \;\; B_2^{P+1} := \{w_2\}, \;\; B_3^{P+1} := Z.$$

Let (c^f, γ^f) denote the modified switched PB-inequality.

3.3.1 Theorem *If* $\mathrm{GRP}(\mathit{\Gamma}, b)$ *is full-dimensional, then the original switched PB-inequality* (c, γ) *is facet-defining for* $\mathrm{GRP}(\mathit{\Gamma}, b)$ *if the modified switched PB-inequality* (c^f, γ^f) *is facet-defining for* $\mathrm{GRP}(\mathit{\Gamma}^f, b)$.

It may be said that the conditions imposed in this theorem are fortunate, since for classical PB-inequalities to define facets the condition that $b(B_j^p : B_{j+1}^p) \geq 2$, for $j = 0, \ldots, n_p$, is sufficient (see 4.4.2).

The proof of Theorem 3.3.1 relates the faces induced by the two inequalities (c, γ) and (c^f, γ^f) geometrically. This relation uses the following mapping and is established in the lemma below. Define the affine mapping

$$h \colon \mathbb{R}^{E(G) \setminus \{f\} \cup \{e_0, e_1, e_2\}} \to \mathbb{R}^{E(G)}$$

by letting, for all $e \in E(G)$,

$$\big(h(y)\big)_e := \begin{cases} 4 - y(\{e_0, e_1, e_2\}), & \text{if } e = f, \\ y_e, & \text{if } e \in E(G) \setminus \{f\}. \end{cases}$$

3.3.2 Lemma *The affine mapping h maps the face of* $\mathrm{GRP}(\mathit{\Gamma}^f, b)$ *induced by the modified switched path-bridge inequality* (c^f, γ^f) *onto the face of* $\mathrm{GRP}(\mathit{\Gamma}, b)$ *induced by the original switched path-bridge inequality* (c, γ).

[1] It may be worth mentioning the similarity of the construction of subdividing an edge twice to the one used by Letchford [Let99] in a different context for the unbounded GRP polyhedron $\mathrm{conv}(\mathscr{S}^\infty)$.

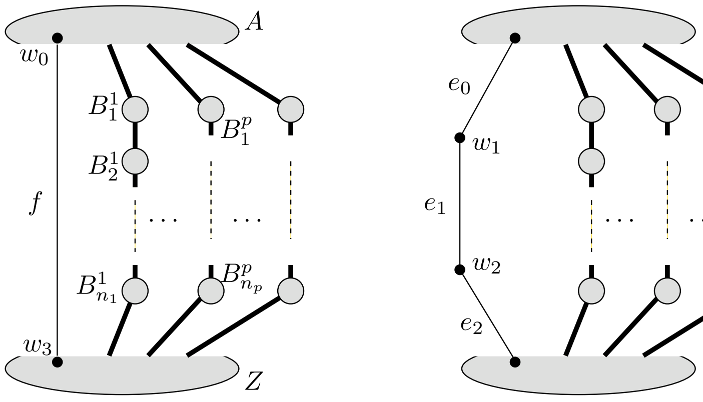


Figure 3.2: Illustration for Theorem 3.3.1.

Proof. First of all we show that for all $y \in \mathbb{R}^{E(G^f)}$, if $x := h(y)$, we have $\gamma - cx = \gamma^f - c^f y$. Note that for all edges in $e \in E(G) \setminus \{f\} = E(G^f) \setminus \{e_0, e_1, e_2\}$ the coefficients of the two inequalities are equal, $c_e = c_e^f$, and $x_e = y_e$ holds, too. For $i = 0, 1, 2$, we have $c_{e_i}^f = 1$. For the right hand sides we have $\gamma^f = \gamma + 4$. From $x_f = 4 - y(\{e_0, e_1, e_2\})$ it follows that

$$\begin{aligned}\gamma - cx &= \gamma - \sum_{e \in E(G)\setminus\{f\}} c_e x_e + x_f \\ &= \gamma - \sum_{e \in E(G^f)\setminus\{e_0,e_1,e_2\}} c_e^f y_e + 4 - y(\{e_0, e_1, e_2\}) = \gamma^f - c^f y.\end{aligned}$$

Now we prove that, if $y \in \mathscr{S}(\Gamma^f, b)$ satisfies the modified inequality with equality, i.e., $c^f y = \gamma^f$, then $x := h(y) \in \mathscr{S}(\Gamma, b)$. We have the following implications:

$$\begin{array}{lcl} y(\{e_0, e_1, e_2\}) = 3 & \Longrightarrow & x_f = 1 \\ y(\{e_0, e_1, e_2\}) = 4 & \Longrightarrow & x_f = 0 \end{array}$$

This would imply $x = h(y) \in \{0, 1\}^{E(G)}$, if no other values of $y(\{e_0, e_1, e_2\})$ occurred. But firstly $y(\{e_0, e_1, e_2\}) \geq 3$ is a consequence of

$$y(\partial(w_1)) \geq 2, \quad y(\partial(w_2)) \geq 2, \text{ and } \quad y(\partial(\{w_1, w_2\})) \geq 2.$$

And secondly, if $y(\{e_0, e_1, e_2\}) = 6$ (the value 5 is impossible for parity reasons), then $c^f y = \gamma$ cannot hold, since the “cheapest” way to connect the remaining sets B_j^p, $p \neq P + 1$ would be to select exactly the two edges with the smallest coefficient out of $\partial(B_j^p)$. This is possible by taking exactly one edge in each of the sets $(B_j^p : B_{j+1}^p)$, $j = 0, \dots, n_p$, which adds up to $\sum_{p \neq P+1} \frac{n_p+1}{n_p-1}$. Even if all edges $g \in F \setminus \{f\}$ have in $y_g = b_g$, it follows that

$$c^f y \geq \sum_{p=1}^{P} \frac{n_p + 1}{n_p - 1} + 6 - b(F \setminus \{f\}) \ > \ \sum_{p=1}^{P} \frac{n_p + 1}{n_p - 1} + 3 + 1 - b(F \setminus \{f\}) = \gamma^f,$$

which proves that $y(\{e_0, e_1, e_2\}) \in \{3, 4\}$ and hence $x \in \{0, 1\}^{E(G)}$.

We come to show that x satisfies the parity constraints. From $y(\{e_0, e_1, e_2\}) = 3$ it follows that $y(e_i) = 1$ for all k, and $y(\{e_0, e_1, e_2\}) = 4$ implies $y(e_i) \in \{0, 2\}$ for all k. Consequently the relation

$$y(\partial(v)) = x(\partial(v)) \mod 2$$

holds for all $v \in V(G)$.

To see that x is a semitour, the connectivity condition remains to be shown. Let S be a union of R-sets in $\mathcal{C}$. We have to show $x(\partial(S)) \geq 2$. If $f \notin \partial(S)$, then $x(\partial(S)) = y(\partial(S)) \geq 2$. Otherwise we can find $S' \subseteq V(G^f)$ with $\partial(S') \setminus \{e_0, e_1, e_2\} = \partial(S) \setminus \{f\}$ and $\partial(S') \cap \{e_0, e_1, e_2\} \neq \emptyset$. From $y(\partial(S')) \geq 2$ it then follows that $x(\partial(S)) \geq 2$.

Finally, it is easy to see that the image of the restriction of h to the face defined by (c^f, γ^f) is just the face defined by (c, γ). This completes the proof of the lemma. □

Now we can tackle the proof of Theorem 3.3.1. Let P be the face of $\mathrm{GRP}(\mathit{\Gamma}, b)$ defined by (c, γ), and let P^f be the facet of $\mathrm{GRP}(\mathit{\Gamma}^f, b)$ defined by (c^f, γ^f). Since, by Lemma 3.3.2, $h(P^f) = P$, and since the dimension of the kernel of the matrix M which defines the affine mapping h is two, we have

$$\dim P = \dim h(P^f) \geq \dim P^f - 2 = |E(G) \setminus \{f\} \cup \{e_0, e_1, e_2\}| - 1 - 2 = |E(G)| - 1.$$

Hence, (c, γ) defines a facet of $\mathrm{GRP}(\mathit{\Gamma}, b)$, if the polyhedron is full-dimensional.

To complete the proof, we give the argument which shows that $\mathrm{GRP}(\mathit{\Gamma}^f, b)$ has full dimension. If we contract the edges e_1, e_2 we again obtain the GRP-structure $\mathit{\Gamma}$, but we have the bound-vector $b' := b + \chi^f$. Clearly, $\mathrm{GRP}(\mathit{\Gamma}, b')$ has full dimension, because it contains $\mathrm{GRP}(\mathit{\Gamma}, b)$. Now it is easy to see that $\mathrm{GRP}(\mathit{\Gamma}^f, b)$ is full-dimensional, too (it also follows from the general Proposition 4.1.8 in the next chapter).

Chapter 4

Dimension and lifting

Ghiani & Laporte [GL00] gave necessary and sufficient conditions under which the polytope $\mathrm{GL}(\varGamma, T)$ is full-dimensional. Further, they investigated the facial structure of the polytope under the condition that it is full-dimensional. They proved necessary and sufficient conditions for the trivial inequalities to define facets of the polytope, and they gave necessary conditions for the facet-defining property of connectivity (1.1b) and cocircuit (2.3) inequalities.[1]

In this chapter we will answer some of the remaining questions about the bounded polyhedra $\mathrm{GRP}(\varGamma, b)$, and obtain results for $\mathrm{GL}(\varGamma, T)$ as consequences. First of all, we prove a node-lifting theorem, which allows us to give sufficient conditions for the facet-defining property of facets of $\mathrm{GRP}(\varGamma, b)$, including connectivity and cocircuit inequalities. Further, we give a complete system of equations for $\mathrm{GRP}(\varGamma, \mathbf{b}^T)$ and $\mathrm{GL}(\varGamma, T)$, and show that if $\mathrm{GRP}(\varGamma, \mathbf{b}^T)$ is not full-dimensional, it is affinely isomorphic to a full-dimensional polyhedron $\mathrm{GRP}(\varGamma/F, \mathbf{b}^T \setminus F)$, where the GRP-structure $\varGamma/F$ can be obtained from $\varGamma$ by edge contractions. Since full-dimensional polyhedra are much easier to study than polyhedra which have a non-trivial space of linear equations, of possibly unknown dimension, our contribution greatly facilitates the further theoretical investigation of polyhedra of this kind.

We introduce some notation which we use in this chapter. First of all we note that in this chapter we will exclusively deal with loopless multigraphs, so all contractions and identifications operate in this class. Let $e = u_1u_2$ be an edge e of a graph G. Let w_e the node of the (loopless multi-) graph G/e which results from contracting e. For a mapping $t\colon V(G) \to \{0,1\}$, we define t/e

$$t/e\colon V(G/e) \to \{0,1\}\colon v \mapsto \begin{cases} t(u_1) + t(u_2) \bmod 2 & \text{if } v = w_e \\ t(v) & \text{otherwise.} \end{cases}$$

For a graph G and a partition $\mathcal{U}$ of its node set, we denote by $G_{\mathcal{U}}$ the graph which results from identifying each set U to a single node. We agree on $V(G_{\mathcal{U}}) = \mathcal{U}$; in other words, the node resulting from identifying the set U is denoted by U. Note that this is the way how we defined $G^{\mathrm{m}}_{\mathcal{C}}$ for a graph G and an R-set partition $\mathcal{C}$. $G_{\mathcal{U}}$

To simplify notation, in this chapter we will restrict to bound vectors $b\colon E(G) \to \mathbb{Z}_+^*$, which means that in this chapters, all edges have finite bounds. Recall the definition of $\lambda_b(G)$ from Section 0.4. We abbreviate $\lambda_{b|_{E(G_{\mathcal{U}})}}(G_{\mathcal{U}})$ to $\lambda_b(G_{\mathcal{U}})$. $\lambda_b(G_{\mathcal{U}})$

4.1 Join structures

Let G, G° be two graphs. In the first part of this section, we will consider partitions of the node set of G° into sets $\{U_v \mid v \in V(G)\} = \mathcal{U}$, with the property that the graph $G^\circ_{\mathcal{U}}$ is isomorphic to G. We introduce the notion of "fatness" as an abstract condition on the structure of the induced

[1] In fact, they conjectured that these conditions were necessary. We were able falsify their conjecture in [RT06], but we do not repeat the construction in this thesis.

subgraphs $G^\circ[U_v]$, which assures that the facet-defining property is inherited. When dealing with node identifications, we have to consider the whole structural information $\mathit{\Gamma} = (G, \mathcal{C}, \mathbf{t})$, and even the bounds b. We also treat "merging" of parallel edges. In the second part, we will give sufficient conditions for fatness, and conjecture a characterization in terms of easy conditions which are both necessary and sufficient.

join structure

Ξ, t-feasible vector

4.1.1 Definition A *join structure* is a triple $\Xi = (H, \mathcal{D}, b)$ consisting of a graph H, a partition $\mathcal{D}$ of $V(H)$, and a vector of (finite) bounds $b\colon E(H) \to \mathbb{Z}_+^*$ on H. Given a join structure Ξ and a parity function t, a vector $x\colon E(H) \to \mathbb{Z}_+$ is called Ξ, t*-feasible,* if

1. it is a t-join,
2. it *connects the sets in $\mathcal{D}$,* i.e., if we delete from the graph $H_{\mathcal{D}}$ all the edges e with $x_e = 0$, then the remaining subgraph is connected, and
3. it satisfies the bounds, that is, $x \le b$.

Clearly, if $(G, \mathcal{C}, \mathbf{t}, T)$ is GL-GRP-structure, then $\Xi := (G, \mathcal{C}, \mathbf{b}^T)$ is a join structure, and $\mathbf{t}$ is a parity function on G. The semitours in $\mathscr{S}(\mathit{\Gamma}, \mathbf{b}^T)$ are precisely the $\Xi, \mathbf{t}$-feasible vectors.

collapsible

fat

4.1.2 Definition We call a join-structure Ξ *collapsible,* if either H has only one node, or for every parity function t on H there exists a Ξ, t-feasible vector. A join-structure Ξ is called *fat,* if either H has only one node, or for every parity function t the set of all Ξ, t-feasible vectors has affine dimension $|E(H)|$, i.e., the affine hull of this set is equal to the full space $\mathbb{R}^{E(H)}$.

Note that fatness implies collapsibility. The term "collapsible" was defined in [Cat88] for the case that $\mathcal{D} = \{\{v\} \mid v \in V(G)\}$.

4.1.3 Remark Suppose that H has at least three nodes and is not two-connected. It is easy to see that the set of all Ξ, t-feasible vectors is the direct product of the feasible vectors on the blocks of H, where the partitions and parity functions for the blocks are defined in a straight forward manner (see 6.1.1). The same relation holds for the affine hulls of the sets of feasible vectors, and also their convex hulls.

The first thing we will study about join structures is what happens if parallel edges are merged. We make precise what we mean by this.

H^*

4.1.4 Definition Let H be a graph, b be a vector of bounds, and $e_1, \dots, e_r$, $r \ge 2$, be parallel edges of H, i.e., $e_1, \dots, e_r \subseteq (u : v)$ for two nodes $u, v \in V(H)$. Suppose that $b_{e_i} = 1$ for $i = 1, \dots, r$. Denote $H^* := H \setminus \{e_2, \dots, e_r\}$ and define bounds b^* by letting

$$b^*_e = \begin{cases} r, & \text{if } e = e_1, \\ b_e, & \text{otherwise.} \end{cases}$$

merging edges

We say that H^* and b^* result from H and b by *merging* the edges $e_1, \dots, e_r$. We say that the join structure Ξ^* arises from a Ξ by merging of edges, if that is the case for the two graphs and the bound vectors and if their node partitions are the same.

The following lemma will be needed below and in Section 4.2.1.

4.1.5 Lemma *Let Ξ^* be the join structure which arises from the join structure Ξ by merging the edges $e_1, \dots, e_r$, and let t be a parity function. Denote by A the affine hull of all Ξ, t-feasible vectors, and by A^* the affine hull of all Ξ^*, t-feasible vectors. Then*

$$\operatorname{codim} A^* \ge \operatorname{codim} A.$$

If $r \ge 3$ or $(H^*, \mathcal{D}, b^* - \chi^{e_1})$ is collapsible, then equality holds. In this case, in particular, $A = \mathbb{R}^{E(H)}$ if and only if $A^* = \mathbb{R}^{E(H^*)}$.

Proof. Consider the linear mapping $f\colon \mathbb{R}^{E(H)} \to \mathbb{R}^{E(H^*)}$ defined by

$$\big(f(x)\big)_e = \begin{cases} x(\{e_1,\dots,e_r\}), & \text{if } e = e_1, \\ x_e, & \text{otherwise.} \end{cases}$$

Clearly f maps A onto A^*. The dimension of the kernel $\ker f$ of f is $r-1$. Hence we have $\dim A^* \geq \dim A - (r-1)$. In this inequality, equality holds if and only if $\ker f$ is contained in the linear space $A - A = \{y - x \mid y, x \in A\}$ define by A. Since $\ker f$ is generated by the vectors $\chi^{e_j} - \chi^{e_{j+1}}$, for $j = 1,\dots,r-1$, it suffices to show that, for each $j = 1,\dots,r-1$, there exist Ξ, t-feasible vectors x, y such that

$$y_{e_j} = 1, \quad x_{e_j} = 0, \qquad y_{e_{j+1}} = 0, \quad x_{e_{j+1}} = 1,$$
$$y_e = x_e, \quad \text{for } e \neq e_j, e_{j+1}.$$

For $r \geq 3$, this is easy. Let $r = 2$. For $j = 1$, the collapsibility of $(H \setminus e_1, \mathcal{D}, b|_{E(H)\setminus e_1})$ implies the existence of an appropriate vector x, and the collapsibility of $(H \setminus e_2, \mathcal{D}, b|_{E(H)\setminus e_2})$ implies the existence of a vector y. □

If $\varGamma^*$ and b^* arise from $\varGamma$ and b by merging, then $\mathrm{GRP}(\varGamma^*, b^*)$ is a projection of $\mathrm{GRP}(\varGamma, b)$. The following lemma gives details on this relationship. It helps in the practical application of Theorem 4.3.1 below, and it also gives a better understanding of the relationship between $\mathrm{GRP}(\varGamma, \mathbf{b}^T)$ and $\mathrm{GL}(\varGamma, T)$.

4.1.6 Lemma *Let $\varGamma = (G, \mathcal{C}, \mathbf{t})$ be a GRP-structure and b a vector of bounds, and suppose that G^*, b^* is obtained by merging a set of parallel edges $\{e_1,\dots,e_r\}$. Define $\varGamma^* := (G^*, \mathcal{C}, \mathbf{t})$. Let $a \in \mathbb{R}^{E(G)}$ with $a_{e_1} = \dots = a_{e_r}$ and $\alpha \in \mathbb{R}$, and define $a^* := a|_{E(G^*)} \in \mathbb{R}^{E(G^*)}$.*

(a). The inequality (a, α) is valid for $\mathrm{GRP}(\varGamma, b)$ if and only if (a^*, α) is valid for $\mathrm{GRP}(\varGamma^*, b^*)$.

(b). If (a, α) is facet-defining for $\mathrm{GRP}(\varGamma, b)$, then (a^*, α) is facet-defining for $\mathrm{GRP}(\varGamma^*, b^*)$.

(c). Suppose that (a^*, α) is facet-defining for $\mathrm{GRP}(\varGamma^*, b^*)$. Then (a, α) is facet-defining for $\mathrm{GRP}(\varGamma, b)$, if

$$\textit{there exists } x \in \mathscr{S}(\varGamma, b) \textit{ with } ax = \alpha \textit{ and } 0 < x(\{e_1,\dots,e_r\}) < r. \qquad (\mathrm{c}^*)$$

If the equation $x_{e_1} = x_{e_2}$ is not valid for the polytope $\mathrm{GRP}(\varGamma, b)$, then condition (c) is also necessary. It is satisfied, for example, if $r \geq 3$, or if e_1 is R-internal and $a_{e_1} \neq 0$, or if $x(\{e_1,\dots,e_r\}) = 2$ is not a valid equation and $a_{e_1} < 0$.*

Proof. The first item is obvious. For item (b), let the hyperplane defined by the equation $c^*x = \gamma$ contain the face of $\mathrm{GRP}(\varGamma^*, b^*)$ induced by (a, α). The vector c^* can be prolonged to a $c \in \mathbb{R}^{E(G)}$ by letting $c_{e_r} := \dots := c_{e_1}$, and then $cx = \gamma$ holds for all x on the face of $\mathrm{GRP}(\varGamma, b)$ defined by (a, α). Hence (c, γ) is a linear combination of (a, α) and valid equations for $\mathrm{GRP}(\varGamma, b)$. Since $c_{e_1} = \dots = c_{e_r}$ and $a_{e_1} = \dots = a_{e_r}$, it follows easily that (c^*, γ) must also be a linear combination of (a^*, α) with a valid equation for $\mathrm{GRP}(\varGamma^*, b^*)$.

For item (c), it is easy to check that the condition (c*) implies that an equation $cx = \gamma$ which defines a hyperplane which contains the face of $\mathrm{GRP}(\varGamma, b)$ defined by (a, α) must satisfy $c_{e_1} = \dots = c_{e_r}$. This implies that $c^*x = \gamma$ holds for all x on the face of $\mathrm{GRP}(\varGamma^*, b^*)$ induced by (a^*, α). Hence (c^*, γ) is a linear combination of (a^*, α) and valid equations for $\mathrm{GRP}(\varGamma^*, b^*)$, which immediately implies that the same holds for (c, γ).

The remaining statements of item (c) are easily verified. □

4.1.1 Characterizing fat join structures

From now on, for the rest of this section, we restrict ourselves to the case that H is 2-connected or has at most two nodes, using Remark 4.1.3. The same argument which is used in [GL00] (see equations (4.2) and (4.3) below) to give necessary conditions for the full-dimensionality of the polytope $\mathrm{GL}(\varGamma, T)$ shows that if a join structure $\Xi = (H, \mathcal{D}, b)$ is fat, then

$$\lambda_b(H) \geq 3, \text{ and} \tag{4.1a}$$

$$\lambda_b(H_{\mathcal{D}}) \geq 4. \tag{4.1b}$$

We will use the following condition for proving sufficient conditions for fatness.

$$H[D] \text{ is connected for each } D \in \mathcal{D}. \tag{4.1c}$$

In the case that H has two nodes, it is easy to check that these conditions are also sufficient. For completeness, we state some facts which are easily verified.

4.1.7 Lemma *Let a join structure $\Xi = (H, \mathcal{D}, b)$ be given.*

(a). If $(H \setminus e, \mathcal{D}, b)$ or $(H, \mathcal{D}, b - \chi^e)$ is fat, then so is Ξ.

(b). For two graphs H^1, H^2, construct the graph H by adding to $H^1 \cup H^2$ a set of edges F each of which having one end node in H^1 and one in H^2. If $\Xi^1 = (H^1, \mathcal{D}^1, b^1)$ and $\Xi^2 = (H^2, \mathcal{D}^2, b^2)$ are fat, and $\Xi = (H, \mathcal{D}^1 \cup \mathcal{D}^2, (b^1, b^2, b_F)^\top)$ where $b_F \in \mathbb{Z}_+^F$ satisfies $\mathbf{1} b_F \geq 4$, then Ξ is fat. □

Now we prove the following proposition, which relates to the special property of GL-GRP-structures, where there are "many" edges with upper bound 2. We define contraction for a join structure $\Xi = (H, \mathcal{D}, b)$. If $e \in E(H)$ has no parallel edge, then define $\Xi/e := (H/e, \mathcal{D}/e, b|_{E(H)\setminus e})$.

4.1.8 Proposition *Let a join structure $\Xi = (H, \mathcal{D}, b)$ be given which satisfies (4.1b) and (4.1c), and let $e \in E(H)$ be an edge with $b_e \geq 2$, which has no parallel edge and its ends are contained in two different sets in $\mathcal{D}$. If Ξ/e is fat, then so is Ξ.*

To prove the proposition, we need the following three lemmas. The first one is well-known. It also allows to derive the condition of Lemma 4.1.5 that $(H^*, \mathcal{D}, b^* - \chi^{e_1})$ be collapsible from (4.1b) and (4.1c). Hence, the condition of Proposition 4.1.8 that e has no parallel edge is only technical and no loss of generality.

4.1.9 Lemma (Folklore) *If a graph G is $2k$-edge connected then for every edge set F with $|F| \leq k$, there exist k edge disjoint spanning trees in $G \setminus F$.*

As a tool, we introduce a way to construct Ξ, t-feasible vectors, which is adapted from the theory of supereulerian graphs [Cat92]. We first note this trivial lemma for easy reference.

4.1.10 Lemma *Let H be a graph, b a vector of bounds, and t a parity function. Let a vector $y \in \mathbb{Z}_+^{E(H)}$ be given which satisfies $y \leq b$. Consider the graph $H(b-y)$, i.e., the spanning subgraph of H with edge set $\{e \in E(H) \mid b_e - y_e > 0\}$. If $H(b-y)$ is connected, then there exists t-join $x \in \mathbb{Z}_+^{E(H)}$ with $y \leq x \leq b$.* □

Suppose that $\mathcal{D}$ is a partition of $V(H)$. We define a "finer" partition $\mathcal{D}^c$ of $V(H)$: the elements of $\mathcal{D}^c$ are the node sets of the connected components of the spanning subgraph H^c of H which results if we delete from H all edges with end nodes in different sets $D \in \mathcal{D}$, i.e., $E(H^c) = E(H) \setminus E(H_{\mathcal{D}}) = \bigcup_{D \in \mathcal{D}} E(D)$. The partition $\mathcal{D}^c$ is finer than $\mathcal{D}$ in the sense that each $D \in \mathcal{D}$ is a union the elements of $D' \in \mathcal{D}^c$ with $D' \subseteq D$. Note that $E(H_{\mathcal{D}^c}) = E(H_{\mathcal{D}})$.

4.1.11 Lemma *Let $\Xi = (H, \mathcal{D}, b)$ be a join structure. Suppose that the following condition holds:*

> *There exist edge sets $T_1, T_2 \subseteq E(H_{\mathcal{D}})$ such that T_1 is a spanning tree in $H_{\mathcal{D}^c}$ and T_2 is a spanning tree in $H_{\mathcal{D}}$, and $\chi^{T_1} + \chi^{T_2} \leq b$.* $(*)$

Then Ξ is collapsible.

Proof. Let t be a parity function. Define $y := \chi^{T_2}$. The graph $H(b - y)$ is connected, because it contains T_1. Hence, Lemma 4.1.10 can be applied and yields a Ξ, t-feasible vector x. $\square$

Proof of Proposition 4.1.8. Let u_1, u_2 be the end nodes of e. Let t be a parity function on H, and let $ax = \alpha$ be an equation which holds for all Ξ, t-feasible vectors x.

Claim. There exists a Ξ, t-feasible vector x^0 with $x^0_e \leq b_e - 2$.

We will construct x^0 below. Now we note that this implies

$$ax^0 = \alpha = a(x^0 + 2\chi^e),$$

and hence $a_e = 0$. Every $\Xi/e, t/e$-feasible vector y can be prolonged to a Ξ, t-feasible vector x with $x_{|G(E)\setminus\{e\}} = y$ in the following way. Since $y(\partial(\{u_1, u_2\})) = t(\{u_1, u_2\}) \bmod 2$, we know that $y(\partial(u_1)) - t(u_1) = y(\partial(u_2)) - t(u_2) \bmod 2$, and we set $x_e = 1$ if this number is odd and $x_e = 2$, if it is even.

But this implies that $a_{|E(H/e)}\, y = \alpha$ holds for all $\Xi/e, t/e$-feasible vectors y. Hence, we have $a = 0$ and $\alpha = 0$.

Proof of the claim. We construct a Ξ, t-feasible vector x^0 with $x^0_e \leq b_e - 2$. By Lemma 4.1.9, we know that there exist two trees, T_1, T_2, such that $\chi^{T_1} + \chi^{T_2} \leq b - 2\chi^e$. By invoking Lemma 4.1.11, we know that there exists a Ξ, t-feasible vector as desired. $\square$

If $|\mathcal{D}| = 1$ and $b = \mathbf{1}$, then the Ξ, t-feasible vectors are precisely the t-joins in H. From [BG86] we know that the convex hull of all t-joins is full-dimensional if the graph is 3-edge connected (4.1a). Using this fact, we obtain the following corollary using Proposition 4.1.8 inductively. The case of $\mathrm{GL}(\Gamma, T)$ occurred in [GL00].

4.1.12 Corollary *Let $(G, \mathcal{C}, \mathbf{t}, T)$ be a GL-GRP-structure. If the conditions (4.1a–b) hold for $(G, \mathcal{C}, \mathbf{b}^T)$, then the polytopes $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$ and $\mathrm{GL}(\Gamma, T)$ are full-dimensional.* $\square$

Finally, we give a result which works in the absence of condition (4.1c).

4.1.13 Proposition *Let a join structure $\Xi = (H, \mathcal{D}, \mathbf{1})$ be given, and let T be the edge set of a spanning tree of $H_{\mathcal{D}}$. If $H \setminus T$ is 3-edge connected then $\Xi = (H, \mathcal{D}, \mathbf{1})$ is fat.*

Proof. The proof is by induction on $|\mathcal{D}|$. For $|\mathcal{D}| = 1$, as already mentioned, the convex hull of t-joins is full-dimensional if the graph is 3-edge connected 4.1a.

Let $|\mathcal{D}| \geq 2$. Suppose that the claim is true for join structures where the partition has at most $|\mathcal{D}| - 1$ sets, and let t be a parity function. Let $f \in T$.

First, we show that there exists a Ξ, t-feasible vector x with $x_f = 0$. Deleting the edge f from the tree with edge set T induces a bipartition of the node set $V(G)$. Let $e \neq f$ be any edge in the cut induced by this bipartition. To define x with $x_f = 0$, we apply Lemma 4.1.10 to $y := \chi^{T\cup\{e\}\setminus\{f\}} = \mathbf{1}$.

Second, we use induction. Define $H' := H \setminus f$. Let u_1, u_2 be the end nodes of f and let $D_1, D_2 \in \mathcal{D}$ such that $u_i \in D_i$, for $i = 1, 2$. Define $\mathcal{D}' := \mathcal{D} \circledcirc f$, and

$$t' \colon V(H') \to \{0, 1\} \colon u \mapsto \begin{cases} 1 - t(u) \bmod 2 & \text{if } u = u_1, u_2 \\ t(u) & \text{otherwise.} \end{cases}$$

Finally let $T' := T \setminus \{f\}$. If $x_{E(H')}$ is a Ξ', t'-feasible vector, then $x_f := 1$ makes x a Ξ, t-feasible vector, hence, by induction, there exist $|E(H)| - 1$ affinely independent Ξ, t-feasible vectors x with $x_f = 1$. Together with at least one feasible vector x with $x_f = 0$, the proposition follows. $\square$

We direct the reader to the possibility to apply Lemma 4.1.5 after this proposition. To complete the picture, we state a conjecture which characterizes fatness.

4.1.14 Conjecture *The obvious necessary conditions for fatness are also sufficient: Every join structure which satisfies (4.1a) and* (4.1b) *is fat.*

4.2 Dimension of and complete systems of equations for GRP polytopes

Now we consider the dimensions of the polytopes $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$ and $\mathrm{GL}(\Gamma, T)$. We note that the codimensions of the two polytopes are equal, and we give a complete system of equations for both polytopes. We show that if $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$ is not full-dimensional, then Γ and T can be modified so that the result is a GL-GRP-structure (Γ', T') with the property that $\mathrm{GRP}(\Gamma', \mathbf{b}^{T'})$ *is* full-dimensional an isomorphic to $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$. As a result, we characterize all facet-defining inequalities of $\mathrm{GL}(\Gamma, T)$ through those of the full-dimensional polytope $\mathrm{GL}(\Gamma', T')$. These results settle the issue of non-full-dimensionality of $\mathrm{GL}(\Gamma, T)$, which was raised in [EL00] who assert that the polytope is “typically not full-dimensional”. An updated statement would be: By contraction of some edges, which can be efficiently identified, and a modification of the cost vector, we can assume without loss of generality that the polytope $\mathrm{GL}(\Gamma, T)$ is full-dimensional.

4.2.1 Constructing an isomorphic full-dimensional GRP-polytope

We will now prove that there exists a set $F \subseteq E(G)$ such that $\mathrm{GRP}(\Gamma/F, \mathbf{b}^{T\setminus F})$ is full-dimensional and

$$\mathrm{GRP}(\Gamma, \mathbf{b}^T) \cong \mathrm{GRP}(\Gamma/F, \mathbf{b}^{T\setminus F})$$

in the sense of affine isomorphism of polytopes. From this we will derive a complete system of valid equations for $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$ and $\mathrm{GL}(\Gamma, T)$, and we show how to gain a complete description of $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$ and $\mathrm{GL}(\Gamma, T)$ out of a complete descriptions for $\mathrm{GRP}(\Gamma/F, \mathbf{b}^{T\setminus F})$ and $\mathrm{GL}(\Gamma/F, T \setminus F)$. We will briefly point out how these results can be used in a preprocessing of GRP-instances to the effect that the optimization takes place over a full-dimensional polytope.

We begin by listing the known valid equations for $\mathrm{GRP}(\Gamma, b)$, which were given in [GL00] for the special case of $\mathrm{GL}(\Gamma, T)$. Because of Remark 4.1.3 (and because the case when G has only two nodes is trivial), for the rest of this section, we assume the case that G is 2-connected, and hence 2-edge connected. For $\{e_1, e_2\} = \partial(U)$ and $b_{e_1} = b_{e_2} = 1$, we have the equations

$$x_{e_1} - x_{e_2} = 0, \qquad \text{if } U \text{ is even, or} \tag{4.2a}$$

$$x_{e_1} + x_{e_2} = 1, \qquad \text{if } U \text{ is odd.} \tag{4.2b}$$

If S is a union of R-sets, and $b(\partial(S)) \leq 3$, then

$$x(\partial(S)) = 2. \tag{4.3}$$

For general bound vectors b there are examples where these equations do not form a complete system. However, for $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$ and $\mathrm{GL}(\Gamma, T)$, they do, as we will show. It will be necessary to treat this issue a bit more general. For the rest of this section, we make the following assumptions on G, $G^{\mathrm{m}}_{\mathfrak{C}}$ and b.

$$\lambda_b(G) \geq 2 \qquad\qquad \lambda_b(G^{\mathrm{m}}_{\mathfrak{C}}) \geq 3 \tag{4.4}$$

$$\begin{aligned} b(\partial(U)) = 2 &\implies |\partial(U)| = 2 && \text{for all node sets } U \\ b(\partial(S)) = 3 &\implies |\partial(S)| = 2 && \text{for all unions of R-sets } S \end{aligned} \tag{4.5}$$

They are satisfied in the case of $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$ but also if we take G^T and merge some of the edge sets $\{e, e^\sim\}$ with $e \in T$. We summarize the main result of this section in the following theorem.

4.2.1 Theorem *There exist a $k \geq 0$ and edges $e_1, \ldots, e_k$ and $f_1, \ldots, f_k$ with the following properties.*

(a). The f_j, $j = 1, \ldots, k$, are distinct and $\{e_1, \ldots, e_k\} \cap \{f_1, \ldots, f_k\} = \emptyset$.

(b). The edge set $D_j := \{e_j, f_j\}$ is a cut in G for $j = 1, \ldots, k$.

(c). For $j = 1, \ldots, k$, we have $b_{e_j} = 1$, and either is f_j R-internal and $b_{f_j} = 1$, or $b_{f_j} = 2$ and $e_j, f_j \in E(G^{\mathrm{m}}_{\mathcal{C}})$.

(d). We abbreviate $F := \{f_1, \ldots, f_k\}$. Either the loopless multigraph G/F consist of a single node, or $E(G/F) = E(G) \setminus F$ and

$$\lambda_{b|_{E(G)\setminus F}}(G/F) \geq 3 \quad \text{and} \quad \lambda_{b|_{E(G)\setminus F}}((G/F)_{\mathcal{C}/F}) \geq 4. \tag{4.6}$$

(e). The following statements are equivalent:

(i) G/F has only one node

(ii) G is a circle with $k+1$ nodes and $E(G) = \{e_1, f_1, \ldots, f_k\}$.

In this case, $\mathrm{GRP}(\mathit{\Gamma}, b)$ is completely described by $0 \leq x_{e_1} \leq 1$ and the equations $c^{(j)}x = \gamma^{(j)}$, $j = 1, \ldots, k$, where

$$(c^{(j)}, \gamma^{(j)}) := \begin{cases} (\chi^{e_j} + \chi^{f_j}, 2), & \text{if } f_j \in E(G^{\mathrm{m}}_{\mathcal{C}}) \\ (-\chi^{e_j} + \chi^{f_j}, 0), & \text{if } f_j \notin E(G^{\mathrm{m}}_{\mathcal{C}}) \text{ and } D_j \text{ even} \\ (\chi^{e_j} + \chi^{f_j}, 1), & \text{if } f_j \notin E(G^{\mathrm{m}}_{\mathcal{C}}) \text{ and } D_j \text{ odd.} \end{cases} \tag{4.7}$$

If G/F has at least two nodes, then the following statements hold.

(f). $\mathrm{GRP}(\mathit{\Gamma}, b)$ is isomorphic to $\mathrm{GRP}(\mathit{\Gamma}/F, b|_{E(G)\setminus F})$.

(g). If $\mathrm{GRP}(\mathit{\Gamma}/F, b|_{E(G)\setminus F})$ is full-dimensional, then every equation $ax = \alpha$ which is valid for $\mathrm{GRP}(\mathit{\Gamma}, b)$ satisfies

$$\binom{a}{\alpha} = \sum_{j=1}^{k} a_{f_j} \binom{c^{(j)}}{\gamma^{(j)}}.$$

Note that the equations (4.7) are trivially linearly independent because of item (a). In the case of the following corollary the condition of item (g) that $\mathrm{GRP}(\mathit{\Gamma}/F, b|_{E(G)\setminus F})$ be full-dimensional is implied by (4.6) because of Proposition 4.1.8.

4.2.2 Corollary *Suppose that (in addition to the conditions (4.4) and (4.5)) there exists an edge set T of a spanning tree of $G^{\mathrm{m}}_{\mathcal{C}}$ such that for all $e \in T$ we have that $b_e \geq 2$ or e has a parallel edge. Then $\dim \mathrm{GRP}(\mathit{\Gamma}, b) = |E(G)| - k$, and the equations of types (4.2) and (4.3) form a complete system of equations for $\mathrm{GRP}(\mathit{\Gamma}, b)$. This holds in particular for $\mathrm{GRP}(\mathit{\Gamma}, \mathbf{b}^T)$.* □

4.2.3 Corollary *We have $\dim \mathrm{GL}(\mathit{\Gamma}, T) = \left|E(G^T)\right| - k$, and the following complete system of equations:*

$$\begin{aligned} x_{e_1} - x_{e_2} &= 0, && \text{for } \{e_1, e_2\} = \partial(U) \text{ with } U \text{ is even} \\ x_{e_1} + x_{e_2} &= 1, && \text{for } \{e_1, e_2\} = \partial(U) \text{ with } U \text{ is odd} \\ x_f + x_{f^\sim} + x_e &= 2 && \text{for } \{f, f^\sim, e\} = \partial(S) \text{ with } S \text{ a union of R-sets.} \end{aligned} \tag{4.8}$$

Proof. It is easy to check the conditions of Lemma 4.1.5 for merging the edge sets $\{e, e^\sim\}$ because of the presence of T and the fact that G is 2-connected. Applying it $|T|$ times, we obtain $\dim \mathrm{GL}(\mathit{\Gamma}, T) = \dim \mathrm{GRP}(\mathit{\Gamma}, \mathbf{b}^T) + |T|$. The result follows from the previous corollary. □

In preparation of the proof of Theorem 4.2.1, we give some structural results.

4.2.4 Lemma *Let $D = \partial(U)$ be given with $b(D) = 2$. Then there exists an R-internal edge $f \in D$. Let $D = \{e, f\}$ with f R-internal. Note that $E(G/f) = E(G) \setminus \{f\}$. The linear projection mapping*

$$\varphi\colon \mathbb{R}^{E(G)} \to \mathbb{R}^{E(G/f)}\colon x \mapsto x|_{E(G)\setminus\{f\}}$$

is a bijection between the sets $\mathscr{S}(\varGamma, b)$ and $\mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}})$, and an isomorphism of the polytopes $\mathrm{GRP}(\varGamma, b)$ *and* $\mathrm{GRP}(\varGamma/f, b|_{E(G)\setminus\{f\}})$.

Proof. The existence of such an $f \in D$ follows by (4.4). Now consider the following affine mapping:

$$\psi\colon \mathbb{R}^{E(G/f)} \to \mathbb{R}^{E(G)}\colon x \mapsto \big(\psi_h(x)\big)_{h\in E(G)}, \text{ where}$$

$$\psi_h(x) := \begin{cases} x_h, & \text{if } h \neq f,\\ x_e, & \text{if } h = f \text{ and } U \text{ is even},\\ 1 - x_e, & \text{if } h = f \text{ and } U \text{ is odd}.\end{cases}$$

We prove that $\varphi\colon \mathscr{S}(\varGamma, b) \to \mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}})$ and $\psi\colon \mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}}) \to \mathscr{S}(\varGamma, b)$ are inverses to each other. Because both φ and ψ are affine mappings, this implies that φ and ψ are isomorphisms of the polytopes.

Let $x \in \mathscr{S}(\varGamma, b)$. Then $x|_{E(G)\setminus\{f\}} \in \mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}})$. Because $x_f = x_e$ if U is even and $x_f = 1 - x_e$ if U is odd, it follows that $x = \psi(x|_{E(G)\setminus\{f\}}) = (\psi \circ \varphi)(x)$.

Now let u, v denote the end nodes of f, where $u \in U$. We show that if $y \in \mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}})$, then $\psi(y) \in \mathscr{S}(\varGamma, b)$. Obviously $\psi(y)$ connects all R-sets, because y does. The parities of all nodes of G, except for, perhaps, u and v, are satisfied by $\psi(y)$. This implies

$$\psi(y)(D) - \mathbf{t}(U) = \sum_{w\in U} \Big(\psi(y)(\partial(w)) - \mathbf{t}(w)\Big) = \psi(y)(\partial(u)) - \mathbf{t}(u) \mod 2$$

and hence the parities of u and v are satisfied if and only if $\psi(y)(D) = \mathbf{t}(U) \bmod 2$. But this is an immediate consequence of the definition of $\psi_f(y)$. Hence $\psi(y) \in \mathscr{S}(\varGamma, b)$.

This completes the proof because the relation $(\varphi \circ \psi)(y) = y$ is obvious. □

4.2.5 Lemma *Let $D = \partial(S)$ with $b(D) = 3$ be given, where S is a union of R-sets. Then there exists an R-external edge $f \in D$ with $b_f = 2$. Let $D = \{e, f\}$ with f R-external and $b_f = 2$. Note that $E(G/f) = E(G) \setminus \{f\}$. Then the linear mapping*

$$\varphi\colon \mathbb{R}^{E(G)} \to \mathbb{R}^{E(G/f)}\colon x \mapsto x|_{E(G)\setminus\{f\}}$$

is a bijection between the sets $\mathscr{S}(\varGamma, b)$ and $\mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}})$, and an isomorphism of the polytopes $\mathrm{GRP}(\varGamma, b)$ *and* $\mathrm{GRP}(\varGamma/f, b|_{E(G)\setminus\{f\}})$.

Proof. It follows from (4.5) that the edge f exists. Define the mapping

$$\psi\colon \mathbb{R}^{E(G/f)} \to \mathbb{R}^{E(G)}\colon x \mapsto \big(\psi_h(x)\big)_{h\in E(G)}, \text{ where}$$

$$\psi_h(x) := \begin{cases} x_h, & \text{if } h \neq f,\\ 2 - x_e, & \text{if } h = f.\end{cases}$$

As in the proof of the previous Lemma, we will show that $\varphi\colon \mathscr{S}(\varGamma, b) \to \mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}})$ and $\psi\colon \mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}}) \to \mathscr{S}(\varGamma, b)$ are inverses to each other, which again implies the isomorphism of the polytopes, because φ and ψ are affine mappings.

For $x \in \mathscr{S}(\varGamma, b)$, we have $x|_{E(G)\setminus\{f\}} \in \mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}})$. Since $b(D) = 3$ and $x(D) \geq 2$, it follows that $x(D) = 2$ and consequently $x_e = 2 - x_f$, and hence $x = (\psi \circ \varphi)(x)$.

Now let u, v denote the end nodes of e, where $u \in S$. We show that if $y \in \mathscr{S}(\varGamma/f, b|_{E(G)\setminus\{f\}})$, then $\psi(y) \in \mathscr{S}(\varGamma, b)$. By an argument similar to the one in the previous lemma the parities are easily seen to be satisfied by $\psi(y)$. For the connectivity condition, let S' be a non-trivial union of R-sets. Since y satisfies the connectivity condition for $G/f, \mathcal{C}/f$, if the cut $\partial(S')$ does not separate the R-sets containing u and v, we have $\psi(y)(\partial(S')) = y(\partial(S')) \geq 2$. But any cut separating u and v contains e, and hence $\psi_e(y) \geq 1$ implies that the connectivity condition is satisfied. It follows that $\psi(y) \in \mathscr{S}(\varGamma, b)$.

Again the proof is finished by the remark that $(\varphi \circ \psi)(y) = y$ is obvious. □

We will have to monitor what happens to valid inequalities or equations. To do this, let $\{e, f\} = D$ be a cut as in Lemma 4.2.4 or 4.2.5. The following lemma is an immediate consequence of these two lemmas if, for any $a \in \mathbb{R}^{E(G)}$ and $\alpha \in \mathbb{R}$, we define $a' \in \mathbb{R}^{E(G/f)}$ and $\alpha' \in \mathbb{R}$ as follows: a'

$$a'_h := \begin{cases} a_h, & \text{if } h \neq e \\ a_e - a_f, & \text{if } h = e \text{ and } D \subseteq E(G^{\mathrm{m}}_{\mathcal{C}}), \\ a_e - a_f, & \text{if } h = e \text{ and } D \text{ is odd}, \\ a_e + a_f & \text{otherwise}; \end{cases} \qquad \alpha' := \begin{cases} \alpha - 2a_f, & \text{if } D \subseteq E(G^{\mathrm{m}}_{\mathcal{C}}) \\ \alpha - a_f, & \text{if } D \text{ odd} \\ \alpha, & \text{otherwise.} \end{cases} \tag{4.9}$$

4.2.6 Lemma *The inequality (a, α) is valid for* $\mathrm{GRP}(\varGamma, b)$ *if and only if (a', α') is valid for* $\mathrm{GRP}(\varGamma/f, b|_{E(G)\setminus\{f\}})$. *Let P be the (possibly trivial) face of* $\mathrm{GRP}(\varGamma, b)$ *induced by (a, α), and P' the (possibly not proper or empty) face of* $\mathrm{GRP}(\varGamma/f, b|_{E(G)\setminus\{f\}})$ *induced by (a', α'). Then φ induces an isomorphism $P \cong P'$ and hence* $\dim P = \dim P'$. □

To use the above lemmas inductively, we only lack one easy technical result.

4.2.7 Lemma *Let f be an edge of G as in Lemma 4.2.4 or 4.2.5. If G is not a triangle, then G/f is still 2-connected.*

Proof. Remember that we assumed that G is 2-connected. Let $\{e, f\} = D$ be the cut in G. Denote by u, v the end nodes of f. To show that G/f is 2-connected, we first treat the case that e and f have a common end node. In this case the common end node must have degree two and G/f is 2-connected if G is not a triangle.

Now suppose that e and f do not have a common end node. There exist precisely two internally node disjoint u, v-paths, namely one consisting of just the edge f and one containing the edge e. Assume that B is a connected component of $G \setminus \{u, v\}$, which does not contain e. There must be neighbors of both u and v in B, because G is 2-connected. But this means that there exists a path in B from a neighbor of u to a neighbor of v. This can be prolonged to a third u, v-path in G, a contradiction. □

Now we are ready to finish the proof of Theorem 4.2.1. It is a well-known fact that every 2-edge cut of G is either part of a circular partition or it is disjoint from all other 2-edge cuts. In a circular partition, there can be at most one R-external edge e with $b_e = 1$. If that is the case, we take all the other edges of the circular partition as the edges $f_1, \ldots, f_l$, and let $e_1 := \ldots := e_l := e$, and if the only edges e with $b_e = 1$ are R-internal, we take for $e, f_1, \ldots, f_l$ all the edges in the circular partition with $b_f = 1$. If a 2-edge cut $\{e, f\}$ is disjoint from every other 2-edge cut, we just let e and f be as in Lemma 4.2.4 or 4.2.5, if the conditions apply. This gives the edges $e_1, \ldots, e_k$ and $f_1, \ldots, f_k$ as in items (a) and (b) of the theorem. Clearly, contracting F either gives a graph with a single node or it produces the condition (4.6). The other statements of the theorem are proved by applying the above Lemmas 4.2.4, 4.2.5, 4.2.6, and 4.2.7 inductively.

4.2.2 Characterization of the facets of non-full-dimensional polytopes

We will show how a complete description of the non-full-dimensional polytope can be gained from a complete description of the one associated with $\varGamma/F$ and $T \setminus F$. In this subsection F will always be the edge set from Theorem 4.2.1, and we assume that G/F has at least two nodes.

4.2.8 Proposition *The facets of* $\mathrm{GRP}(\Gamma, b)$ *are precisely the faces induced by inequalities* (a, α) *with the property that* $(a|_{E(G\setminus F)}, \alpha)$ *is facet defining for* $\mathrm{GRP}(\Gamma/F, b|_{E(G\setminus f)})$ *and* $a|_F = \mathbf{0}$.

Proof. Let $f \in F$ and let $\bar{a} \in \mathbb{R}^{E(G\setminus f)}$, $\alpha \in \mathbb{R}$ be given. Then, if we define $a \in \mathbb{R}^{E(G)}$ by $a|_{E(G\setminus f)} := \bar{a}$ and $a_f := 0$, we have $a' = \bar{a}$ and $\alpha' = \alpha$, where a' and α' as in (4.9). Hence (a, α) is facet-inducing if and only if $(\bar{a}, \alpha)$ is. Repeated application of this fact yields the claim. □

4.2.9 Corollary *A facet of* $\mathrm{GL}(\Gamma, T)$ *is induced either by an upper-bound inequality*

$$x_f \leq 1 \quad \text{or} \quad x_{f^\sim} \leq 1, \qquad \text{where } f \in F \cap T, \tag{4.10}$$

or by an inequality (a, α) *with the property that* $(a|_{E(G\setminus F)}, \alpha)$ *is facet defining for* $\mathrm{GL}(\Gamma/F, T \setminus F)$ *and* $a|_F = \mathbf{0}$. *All inequalities of the latter kind are facet-inducing for* $\mathrm{GL}(\Gamma, T)$.

Proof. Define $\tilde{F} := \{f^\sim \mid f \in F \cap T\}$. Let $\hat{G} := G^T \setminus \tilde{F}$, and define $b := \mathbf{1} + \chi^{F\cap T}$. Then G and b satisfy conditions (4.4) and (4.5). Let $\hat{\Gamma} := (\hat{G}, \mathcal{C}, \mathbf{t})$.

Claim. The projection mapping $a \mapsto a|_{E(G)\setminus\tilde{F}} : \mathbb{R}^{E(G^T)} \to \mathbb{R}^{E(\hat{G})}$ defines a bijection between the set of facet-defining inequalities of $\mathrm{GL}(\Gamma, T)$ which are not equivalent to an upper-bound inequality (4.10) and the set of facet-defining inequalities of $\mathrm{GRP}(\hat{\Gamma}, b)$.

Once this claim is proven, the statement immediately follows from that of the previous proposition.

Proof of the claim. Let $\{e, f\} = D$ be as in Lemma 4.2.5, i.e., $f \in T$. We will prove that every inequality (a, α) which is valid for $\mathrm{GL}(\Gamma, T)$ but which does not satisfy $a_f = a_{f^\sim}$ is dominated by or equivalent to an upper-bound inequality. Once we know that $a_f = a_{f^\sim}$ holds for an inequality (a, α) which is facet-defining for $\mathrm{GL}(\Gamma, T)$, we conclude that $(a|_{E(G\setminus f^\sim)}, \alpha)$ is facet-defining for $\mathrm{GRP}(\hat{\Gamma}, b)$ by Lemma 4.1.6-(b).

Assume $a_f < a_{f^\sim}$. If x lies on the face, then we have the following implication

$$x_{f^\sim} = 1 \Rightarrow x_f = 1 \Rightarrow x_e = 0.$$

Hence the equation $x_{f^\sim} + x_e = 1$ is valid for all x on the face. This equation is equivalent to $x_f = 1$. Now either the inequality (a, α) is dominated by or equivalent to $x_f \leq 1$, or the equation $x_f = 1$ is valid for the polytope. But the latter case cannot occur, because then $(\chi^f, 1)$ would be a linear combination of equations (4.8), each of which satisfies $c_f = c_{f^\sim}$.

It remains to show that (a, α) is facet inducing if $(a|_{E(G\setminus f^\sim)}, \alpha)$ is. This is a consequence of Lemma 4.1.6, since condition (c*) of is a consequence of the fact that $x_e = 0$ is not a valid equation for $\mathrm{GRP}(\hat{\Gamma}, b)$, because of Theorem 4.2.1-(g). □

The following proposition completes the picture. It shows that the inequalities with $a|_F = \mathbf{0}$ have a "configuration-like" form, in the sense that there exists a partition $\mathcal{U}$ of $V(G)$, such that $F = \bigcup_U E(U)$ for a $U \in \mathcal{U}$.

4.2.10 Proposition *If the case (e) of Theorem 4.2.1 is not given, the connected components of the graph with node set* $V(G)$ *and edge set* $F := \{f_1, \ldots, f_k\}$ *are induced subgraphs of* G.

Proof. If the end nodes of an edge e are connected by a path P using only edges in F, then e forms a cut with each of the edges of the path. Hence, if G is 2-connected it must be a circle as in (e)ii. But this case is excluded in this subsection. □

4.3 Lifting

4.3.1 0-Node lifting

Now we give the intended result on 0-node lifting for $\mathrm{GRP}(\Gamma, b)$. For this, we use the definition for 0-node lifting as described in (2.4) on page 20. For a subset $U \subseteq V(G)$, we define $\mathcal{C} \cap U$ to be the partition of U consisting of sets $C' \subseteq U$ with $C' = C \cap U$ for a $C \in \mathcal{C}$. We have the following fact.

4.3.1 Theorem *Suppose that for each $U \in \mathfrak{U}$ the join structure $\Xi_U := (G^\circ[U], \mathcal{C} \cap U, b^\circ|_{E(U)})$ is fat. If (a,α) defines a non-empty face of codimension r of $\mathrm{GRP}(\mathit{\Gamma}, b)$, then the inequality (a°, α) obtained by 0-node lifting (a,α) defines a face of codimension r of $\mathrm{GRP}(\mathit{\Gamma}^\circ, b^\circ)$.*

Proof. Let $x \in \mathscr{S}(\mathit{\Gamma}, b)$. Define $x^\circ \in \mathbb{Z}_+^{E(G^\circ)}$ by first letting $x^\circ|_{E(G)} := x|_{E(G)}$. Then define for each $U \in \mathfrak{U}$ with $|U| > 1$ a parity function t_U by

$$t_U(v) := t(v) + x^\circ(\partial(v)) \mod 2 \qquad \text{for each } v \in U.$$

Finally, for each $U \in \mathfrak{U}$ with $|U| > 1$, let $x^\circ|_{E(U)}$ be a Ξ_U, t_U-feasible vector, which is possible because Ξ_U is collapsible. Clearly $x^\circ \in \mathscr{S}(\mathit{\Gamma}^\circ, b^\circ)$ and $a^\circ x^\circ = ax$ hold.

Now let P be the face of $\mathrm{GRP}(\mathit{\Gamma}, b)$ induced by $ax = \alpha$ and let P° be the face of $\mathrm{GRP}(\mathit{\Gamma}^\circ, b^\circ)$ induced by $a^\circ x = \alpha$. Writing down all the semitours $x^\circ \in \mathscr{S}(\mathit{\Gamma}^\circ, b^\circ)$ which can be obtained by the above construction, the fact that all the Ξ_U are fat implies that

$$\dim \mathrm{GRP}(\mathit{\Gamma}^\circ, b^\circ) \geq \dim \mathrm{GRP}(\mathit{\Gamma}, b) + \sum\nolimits_{U \in \mathfrak{U}} |E(U)| \quad \text{and}$$
$$\dim P^\circ \geq \dim P + \sum\nolimits_{U \in \mathfrak{U}} |E(U)|.$$

But the inequality '$\leq$' is trivial for both the polytopes and the faces. Combining the two equations using $\dim P = \dim \mathrm{GRP}(\mathit{\Gamma}, b) - r$, we see that $\dim P^\circ = \dim \mathrm{GRP}(\mathit{\Gamma}^\circ, b^\circ) - r$. □

The graph G in the theorem may have a big number of parallel edges. This can be remedied by merging edges, see Def. 4.1.4 on page 34.

4.3.2 Parallel edges

We now give some results about lifting on single edges. This is necessary since facet-defining inequalities for GRP-polyhedra with bounds do not necessarily have configuration form (see 2.2.2).

4.3.2 Proposition *Let $\mathit{\Gamma} = (G, \mathcal{C}, \mathbf{t})$ be a GRP-structure and suppose that $\mathrm{GRP}(\mathit{\Gamma}, b)$ has full dimension. Let (a, α) be a facet-defining inequality for $\mathrm{GRP}(\mathit{\Gamma}, b)$ which is not equivalent to an inequality of the form $x_e \geq 0$ or $x_e \leq b_e$. Let u,v two nodes in the same R-set with $(u : v) \neq \emptyset$, such that $b_e = 1$ for all $e \in (u : v)$. Suppose G' is obtained from G by adding an edge $f \notin E(G)$ with end nodes u and v to G, Let $\mathit{\Gamma}' := (G', \mathcal{C}, \mathbf{t})$ and b' a vector of bounds on the edges of G' with $b|_{E(G)} = b$ and $b_f = 1$. Define $a'_e := a_e$ for all $e \in E(G) \subseteq E(G')$ and, fixing a $g \in (u : v)$, let*

$$a'_f := |a_g|.$$

The inequality (a', α) is facet-defining for $\mathrm{GRP}(\mathit{\Gamma}', b')$.

We note that, by Lemma 4.1.6, the condition $b_e = 1$ for all $e \in (u : v)$ is no restriction to generality.

Proof. For the validity, let x be a semitour in $\mathscr{S}(\mathit{\Gamma}', b')$ with $x_f = 1$. If $x_g = 0$, then $a'x \geq a'(x - \chi^f + \chi^g) \geq \alpha$. Otherwise, we have $a'x \geq a'(x - \chi^f - \chi^g) \geq \alpha$.

For the facet-defining property, we need to show that there exists a semitour x with $x_f = 1$ satisfying $a'x = \alpha$. Suppose first that $a_f = a_g$. Since the equality $x_g = 0$ does not hold for all the semitours x in $\mathscr{S}(\mathit{\Gamma}, b)$ satisfying $ax = \alpha$, we can select a semitour x such that $x_g = 1$. Now $x - \chi^g + \chi^f$ is a semitour and satisfies $a'x = \alpha$. In the case that $a_f = -a_g$, take a semitour satisfying $x_g = 0$ and use $x + \chi^g + \chi^f$. □

To deal with negative coefficients in the case of Proposition 4.3.2, we refer to Section 3.1. If the nodes u and v are contained in distinct R-sets, we have similar results.

4.3.3 Proposition *Suppose that $\mathrm{GRP}(\mathit{\Gamma}, b)$ has full dimension and let (a, α) be a facet-defining inequality for $\mathrm{GRP}(\mathit{\Gamma}, b)$, which is not equivalent to an inequality of the form $x_e \geq 0$ or $x_e \leq b_e$. Let u,v two nodes in distinct R-set C with $|(u : v)| \geq 2$ and $b_e = 1$ for all $e \in (u : v)$.*

(a). Suppose $\varGamma'$ is obtained from $\varGamma$ by replacing the graph G by the graph G' which results from G by adding an edge $f \notin E(G)$ with end nodes u and v to G. Let $b'|_{E(G)} = b$ and $b'_f = 1$. Define $a'_e := a_e$ for all $e \in E(G) \subseteq E(G')$ and

$$a'_f := \max_{e\in(u:v)} |a_e|.$$

The inequality (a', α) is facet-defining for $\mathrm{GRP}(\varGamma', b)$.

(b). Suppose that there are edges $g, \breve{g} \in (u : v)$ such that $a_g < 0$ and $a_g \le -a_{\breve{g}}$ and that G' is obtained from G by adding two edges $f_1, f_2 \notin E(G)$ with end nodes u and v to G. Denote the resulting GRP-structure by $\varGamma'$. Again, let $b'|_{E(G)} = b$ and $b'_{f_1} = b'_{f_2} = 1$. Define $a'_e := a_e$ for all $e \in E(G) \subseteq E(G')$, and

$$a'_{f_1} := a'_{f_2} := a_g.$$

The inequality $(a', \alpha + 2a_g)$ is facet-defining for $\mathrm{GRP}(\varGamma', b)$.

Proof. (a). For the validity, assume that $x \in \mathscr{S}(\varGamma', b')$ with $x_f = 1$. If $x(u : v) \le 2$, then there exists an edge $g \in (u : v)$ with $x_g = 0$, thus we have $a'x \ge a'(x - \chi^f + \chi^g) \ge \alpha$. If $x(u : v) \ge 3$, then taking away to edges in $(u : v)$ from the semitour does not endanger connectivity, so that, if g is any edge in $(u : v) \setminus f$ with $x_g = 1$, we have $a'x \ge a'(x - \chi^f - \chi^g) \ge \alpha$.

The facet defining property is proved in the same way as in the previous proposition.

(b). For the validity, let us first consider all semitours $x \in \mathscr{S}(\varGamma', b')$ with $x_{f_1} = x_{f_2} = 1$. If $y := x - \chi^{\{f_1,f_2\}}$ satisfies the connectivity constraints, then it is a semitour on G and we have $ax = ay + 2a_g \ge \alpha + 2a_g$. If that is not the case, then $y_g = y_{\breve{g}} = 0$ and $z := y + \chi^{\{g,\breve{g}\}}$ is a semitour on G, which satisfies $az \ge \alpha$ and consequently, because $a_g + a_{\breve{g}} \le 0$, we have $a'x = ay + 2a_g \ge az + 2a_g \ge \alpha + 2a_g$.

Now, consider semitours with $x_{f_1} = 1$ and $x_{f_2} = 0$ (w.l.o.g.). If $x_g = 0$, we can replace x by $y := x - \chi^{f_1} + \chi^g$, and obtain $ax = ay \ge \alpha \ge \alpha + 2a_g$. Otherwise, if $x_g = 1$, we replace x by $x + \chi^{f_2} - \chi^g$, which results in the case discussed above.

For the facet-defining property, take $|E(G)|$ affinely independent semitours in $\mathscr{S}(\varGamma, b)$ satisfying the inequality (a, α) with equality and add $\chi^{\{f_1,f_2\}}$ to each of them to obtain semitours in $\mathscr{S}(\varGamma', b')$. Then, since $x_g = 1$ is not a valid equation for the facet defined by (a, α), we can take a semitour $x \in \mathscr{S}(\varGamma, b)$ satisfying $x_g = 0$ and $ax = \alpha$. The two vectors $y^1 := y + \chi^g + \chi^{f_1}$ and $y^2 := y + \chi^g + \chi^{f_2}$ are semitours on G'. We have constructed $|E(G')| = |E(G)| + 2$ affinely independent semitours on G' satisfying the inequality $(a', \alpha + 2a_g)$ with equality. □

4.4 Facet-defining property for the polyhedra with bounds

In this section we will apply Theorem 4.3.1 and Lemma 4.1.6 to prove sufficient conditions for some valid inequalities to define facets of $\mathrm{GRP}(\varGamma, \mathbf{b}^T)$ and $\mathrm{GL}(\varGamma, T)$. Unless otherwise stated, from now on for the rest of this section, we assume that G is 2-connected or has only two nodes. As we have seen in the previous section, the assumption that the polytope is full-dimensional is not a restriction in the case of the polytopes $\mathrm{GRP}(\varGamma, \mathbf{b}^T)$ and $\mathrm{GL}(\varGamma, T)$. Hence, from now on, we will assume that $\mathrm{GRP}(\varGamma, b)$ is full-dimensional. Let H be a subgraph of G. We informally say that H is fat, if the join structure $\Xi := (H, \mathcal{C} \cap V(H), b|_{E(H)})$ is fat.

4.4.1 Connectivity and blossom facets

4.4.1 Proposition *Let $\varGamma$ be a GRP-structure and $b\colon E(G) \to \mathbb{Z}_+^* \cup \{\infty\}$ a vector of bounds.*

(a). A connectivity inequality (1.1b) *defines a facet of* $\mathrm{GRP}(\varGamma, b)$, *if* $b(\partial(S)) \ge 4$ *and both shores of the cut are fat.*

(b). A blossom inequality (2.5a)

$$x(\partial(U) \setminus F) - x(F) \ge 1 - b(F), \quad \textit{where } \mathbf{t}(U) + b(F) \textit{ is odd,}$$

defines a facet of $\mathrm{GRP}(\varGamma, b)$*, if both shores are fat and* $b(F) \geq 3$ *holds in the case that* U *is a union or R-sets.*

Proof. By Theorem 4.3.1, we may assume that the shores of the cuts are single nodes. Clearly, the inequality defines a facet if G has a single edge e with $b_e \geq 4$. The result follows from applying Lemma 4.1.6-(c), which is possible because $r \geq 3$, and then Lemma 4.1.6-(b) for the edges $e \in \partial(S)$ with $b_e \geq 2$.

The proof for the blossom inequalities is similar, except that it requires a number of merely technical case distinctions. □

4.4.2 Path-bridge facets

Now we deal with the path-bridge inequalities, see 2.1.2.

4.4.2 Proposition *Suppose that* $\mathrm{GRP}(\varGamma, b)$ *is full-dimensional. A path-bridge inequality defines a facet of* $\mathrm{GRP}(\varGamma, b)$*, if each of the induced subgraphs* $G[A]$*,* $G[Z]$*,* $G[B^p_j], \dots, G[B^P_{n_P}]$ *is fat and* $b(B^p_j : B^p_{j+1}) \geq 2$ *for each* $j = 0, \dots, n_p$*,* $p = 1, \dots, P$*.* □

Proof. After invoking Theorem 4.3.1 and restricting to the case that each of the edge sets $(B^p_j : B^p_{j+1})$, $j = 0, \dots, n_p - 1$, consists of a single edge e (by Lemma 4.1.6), the proof of the following proposition is essentially a combination of those in [CFN85] and [CS94] for $\mathrm{GRP}(\varGamma, \infty)$. □

We also give a result in the opposite direction: which path-bridge inequalities do *not* define facets of the polyhedra with bounds. The proposition and its proof are generalizations of an observation of Letchford [Let03].

4.4.3 Proposition *Suppose that in a path-bridge configuration, there exists a path consisting of only two beads* B^p_1*,* B^p_2*. If* $b(B^p_1 : B^p_2) \leq 1$*, then the (switched) path-bridge inequality does not define a facet of* $\mathrm{GRP}(\varGamma, b)$*.*

Proof. For the case $b(B^p_1 : B^p_2) = 1$, it is easily checked that the (switched) path-bridge inequality is dominated by the connectivity inequalities for the sets B^p_1 and B^p_2 and the switched path-bridge inequality which results if the pth path is replaced by a flipped edge. The case that $b(B^p_1 : B^p_2) = 0$ is trivial. □

4.4.3 Honeycomb facets

We come to the honeycomb inequalities. In addition to the conditions in 2.1.2 on page 18, the following condition is required to hold:

6. If in the tree $\mathfrak{T}$ each node set $\{B^k_1, \dots, B^k_{n_k}\}$ is identified to a single node B^k, the resulting graph must be 2-connected.

4.4.4 Proposition *Suppose that* $\mathrm{GRP}(\varGamma, b)$ *is full-dimensional. A honeycomb inequality defines a facet of* $\mathrm{GRP}(\varGamma, b)$*, if each of the induced subgraphs* $G[B^i_j]$ *is fat and* $b(B : B') \geq 2$ *for each edge* $\{B, B'\}$ *of the tree* $\mathfrak{T}$*.*

Proof. After invoking Theorem 4.3.1 and restricting to the case that each of the edge sets $(B : B')$ for each edge $\{B, B'\}$ of the tree $\mathfrak{T}$ consists of a single edge e with $b_e \geq 2$, the proof is completely analogous to the one in [CS98] for $\mathrm{GRP}(\varGamma, \infty)$.

However, the application of Lemma 4.1.6 is non-trivial. The condition (c*) must be proved for edges in $(B : B')$ for $\{B, B'\}$ in the tree $\mathfrak{T}$. This can be done by a nice inductive argument.

Suppose that $B^k_j = \{v^k_j\}$. Let e be an edge in any of the sets $(v^k_{j_1} : v^k_{j_2})$, with $j_1 \neq j_2$. Consider the path P in T from $B^k_{j_1}$ to $B^k_{j_2}$; let $g_1, \dots, g_q$ be the set of its edges. Because the nodes B^k_j, for $k = 1, \dots, L$, have degree 1 in T (condition 1 in the description of HC-inequalities in Section 2.1.2), all except the first and the last nodes of P are in $\{B^{k'} \mid k' = L + 1, \dots, K\}$. This means that

$P^\circ := P \setminus \{B^k_{j_1}, B^k_{j_2}\}$ can be viewed as a path in $T_A \setminus B^k$. Because T_A is two-connected, $T_A \setminus B^k$ is connected, and thus we can find a spanning tree in $T_A \setminus B^k$ containing P°. Consequently, we have K edges $g_1, \ldots, g_K$ (with $g_1, \ldots, g_q$ from above) connecting all the sets $\bigcup_j B^{k'}_j$, $k' = 1, \ldots, L$ and $B^{k'}_1$, $k = L+1, \ldots, K$. If we chose $x_e = 1$, $x_{f_1(g_j)} = 1$ for $j = 1, \ldots, K$ and $x_{f_2(g_j)} = 1$ for $j = q+1, \ldots, K$, we have constructed a semitour x which satisfies the HC-inequality with equality: $cx = 2(K-1)$.

What we will show now, is that for each edge g of T there exist $B^k_{j_1}$ and $B^k_{j_2}$ with $(v^k_{j_1} : v^k_{j_2}) \neq \emptyset$, such that g lies on the path in T connecting the nodes $B^k_{j_1}$ and $B^k_{j_2}$ of T. The following argument proves this statement.

Let g be an edge in T. It induces a cut in T. Because T_A is two-edge-connected, there exists an $k \in \{1, \ldots, L\}$ such that identifying the nodes $B^k_1, \ldots, B^k_{n_k}$ of T to the node B^k of T_A connects the two shores of the cut. Thus there exist nodes B^k_i and B^k_j in T such that g lies on the path P connecting them in T. But we require a pair such that $(v^k_i : v^k_j) \neq \emptyset$. We use an induction argument. Because of the condition 5 of the definition of Honeycomb configurations in 2.1.2, there exists a path Q in the induced subgraph G^k defined above, which connects the nodes i and j of G^k. The induction is on the length of Q. If the length of Q is one, we are finished. Otherwise, let i' be the node incident to i on Q. There exists a path $\bar{P}$ in T connecting $B^k_{i'}$ to B^k_i. If g lies on that $\bar{P}$, we take $\bar{P}$ instead of P and we are finished. In the other case, because B^k_i has degree one in T, the two paths P and $\bar{P}$ meet at a node different from B^k_i. We replace P by the path which connects $B^k_{i'}$ and B^k_j in $P \cup \bar{P}$, and Q by the sub-path of Q connecting i' and j. Thus, by induction, we have proved that there exists a path P such that the nodes $v^k_{j_1}$ and $v^k_{j_2}$ are connected by an edge in G. □

Without any other than a technical difficulty, the same proof can be adapted to switched honeycomb inequalities as defined in 3.3.1.

4.4.4 New facets

We now introduce two new types of valid inequalities. The key idea occurs in [RT06]. It is unlikely that they are of any practical importance, but they are noteworthy for three reasons. First, although they have non-negative coefficients, they are not valid for $\mathrm{GRP}(\Gamma, \infty)$. Second, they are not configuration inequalities (see 2.1.1), as all non-trivial facet defining inequalities of $\mathrm{GRP}(\Gamma, \infty)$ are. The third reason why we give these inequalities (4.11) and (4.12) is that the proof of the facet-defining property of the inequality (4.12) can be done by an intriguing non-standard lifting argument based on Sections 4.1 and 4.3, but not on Theorem 4.3.1. In other words, inequality (4.12) can be obtained by "lifting" the ostensibly completely different inequality (4.11) in a new way.

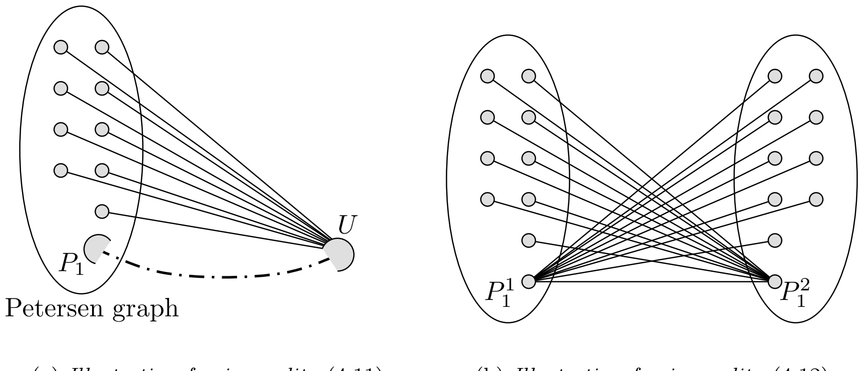
Let a partition of $V(G)$ into sets $P_1, \ldots, P_{10}$, U be given. Denote this partition by $\mathcal{U}$ and suppose that the following properties hold (see Fig. 4.1(a)):

1. The sets U and P_1 are odd.
2. The loopless multigraph $G_{\mathcal{U}} - U$ is isomorphic to the Petersen graph, and $b_e = 1$ for the 15 edges of this graph. Note that, by our convention, $U \in V(G_{\mathcal{U}})$.
3. $b(P_1 : U) \geq 2$, and if equality holds, then there is only one edge in $(P_1 : U)$.
4. $b(P_j : U) \geq 1$ for $j \neq 1$.

The inequality

$$x\big(P_2 \cup \cdots \cup P_{10} : U\big) \geq 1 \tag{4.11}$$

is valid for $\mathrm{GRP}(\Gamma, b)$. This is so because, if $x\big(P_2 \cup \cdots \cup P_{10} : U\big) = 0$ then, to achieve connectivity in $H := G_{\mathcal{U}} - U$, x must contain a spanning Eulerian subgraph of H. Since H is the Petersen graph, which is 3-regular and not Hamiltonian, this cannot be the case.



(a) *Illustration for inequality* (4.11). (b) *Illustration for inequality* (4.12).

Figure 4.1: Two new types of facet defining inequalities for $\mathrm{GRP}(\mathit{\Gamma}, b)$

4.4.5 Proposition *If* $\mathrm{GRP}(\mathit{\Gamma}, b)$ *is full-dimensional, and the induced subgraphs* $G[P_1], \ldots, G[P_{10}]$, $G[U]$ *are fat, then the inequality* (4.11) *is facet-defining for* $\mathrm{GRP}(\mathit{\Gamma}, b)$. □

Proof. The statement can be proved by enumeration after shrinking the sets $P_1, \ldots, P_{10}$, and U, using Theorem 4.3.1. □

The proposition is not true if $b(P_1 : U) = |(P_1 : U)| = 2$. This is an example where Lemma 4.1.6 cannot be applied. Now we come to the second new type of valid inequalities. Let a partition of $V(G)$ into sets $P_1^1, \ldots, P_{10}^1$, $P_1^2, \ldots, P_{10}^2$ be given. Denote $\mathcal{U} := \{P_1^1, \ldots, P_{10}^1, P_1^2, \ldots, P_{10}^2\}$, and suppose that the following conditions hold (see Fig. 4.1(b), the edges in the Petersen graphs are omitted in the figure):

1. Each of the sets P_j^i is a union of R-sets.
2. The subgraph of the loopless multigraph $G_{\mathcal{U}}$ induced by the nodes $P_1^i, \ldots, P_{10}^i$ is isomorphic to the Petersen graph for $i = 1, 2$. We have $b_e = 1$ for each edge $\{e\} = (P_j^i : P_k^i)$, $i = 1, 2$, $j, k = 1, \ldots, 10$.
3. $(P_j^1 : P_k^2) = \emptyset$ if $j \neq 1 \wedge k \neq 1$.
4. $b(P_k^1 : P_1^2) \geq 2$ and $b(P_1^1 : P_k^2) \geq 1$ for all $k = 1, \ldots, 10$.
5. $b(P_1^1 : \bigcup_{k \geq 2} P_k^2) \geq 12$.

By the same argument as above for inequality (4.11), we see that the following inequality is valid for $\mathrm{GRP}(\mathit{\Gamma}, b)$:

$$x\big(P_2^1 \cup \cdots \cup P_{10}^1 \ : \ P_1^2\big) \geq 1. \tag{4.12}$$

4.4.6 Proposition *If* $\mathrm{GRP}(\mathit{\Gamma}, b)$ *is full-dimensional and the induced subgraphs* $G[P_j^i]$, $i = 1, 2$, $j = 1, \ldots, 10$ *are fat, then the inequality* (4.12) *is facet-defining for* $\mathrm{GRP}(\mathit{\Gamma}, b)$.

Proof. First, we invoke Theorem 4.3.1, i.e., w.l.o.g., we assume that the node sets are singletons $P_j^i = \{p_j^i\}$. It can easily be verified that Lemma 4.1.6-(c*) is applicable, even in the case when $b(P_1^1 : P_1^2) = \big|(P_1^1 : P_1^2)\big| = 2$.

We will invoke Proposition 4.4.5. Consider the graphs $H^1 := G[\{p_1^1, \ldots, p_{10}^1, p_1^2\}]$ and $H^2 := G[\{p_1^1, p_1^2, \ldots, p_{10}^2\}] \setminus (p_1^1 : p_2^1)$, see Fig. 4.2. Assign parities t^1, t^2 to the nodes of H^1 and H^2, respectively, by letting $t^i(u) := 1$ if $u \in \{p_1^1, p_1^2\}$, and $t^i(u) := 0$ otherwise. For $i = 1, 2$, we define a partition of $V(H^i)$ by $\mathcal{C}^i := \big\{\{p_1^i, p_1^{2-i}\}, \{p_2^i\}, \ldots, \{p_{10}^i\}\big\}$. Now we have a GRP-structure

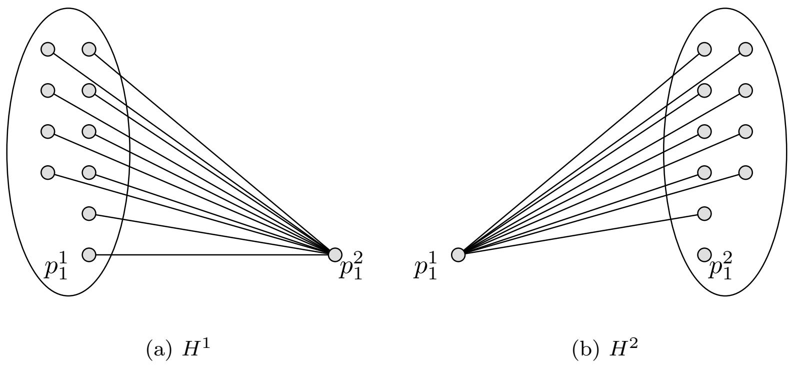


Figure 4.2: Illustration for the proof of Proposition 4.4.6

$\mathit{\Gamma}^1 := (H^1, \mathcal{C}^1, t^1)$ and a join structure $\Xi^2 := (H^2, \mathcal{C}^2, b|_{E(H^2)})$. The crucial observation now is this: if $x \in \mathscr{S}(\mathit{\Gamma}^1, b|_{E(H^1)})$ and y is a Ξ^2, t^2-feasible vector, then $\binom{x}{y} \in \mathscr{S}(\mathit{\Gamma}, b)$.

Proposition 4.4.5 states that there exist $|E(H^1)|$ affinely independent vectors in $\mathscr{S}(\mathit{\Gamma}^1, b|_{E(H^1)})$, which satisfy $x(\{p_2^1, \dots, p_{10}^1\} : p_1^2) = 1$. If we can prove that there exist $|E(H^2)| + 1$ affinely independent Ξ^2, t^2-feasible vectors, then the proposition is true. But we apply Proposition 4.1.13, to show that Ξ^2 is fat. Namely, we choose

$$T := \left(p_1^1 : \{p_2^2, \dots, p_{10}^2\}\right) = \partial_{H^2}(p_1^1)$$

as the edge set of a tree. Because of condition 5 above and the fact that the Petersen graph is 3-edge connected, Proposition 4.1.13 is applicable and the proof is completed. □

Chapter 5

A focus on the Graphical Traveling Salesman Polyhedron

As mentioned before, the Graphical Traveling Salesman Problem is a special case of the General Routing Problem, and the same holds for the associated polyhedra. In this chapter we particularly address the Graphical Traveling Salesman Polyhedron. For ease of notation, we abbreviate GTSP(n) := GTSP(K_n) := GRP($\mathit{\Gamma}, \infty$), with $\mathit{\Gamma} := (K_n, V(K_n), \mathbf{0})$, and we denote $V_n := V(K_n)$ and $E_n := E(K_n)$. We will also fix $m := |E_n| = \binom{n}{2}$.

GTSP(n), V_n, E_n, m

There are three reasons for focusing on the special case. The first reason is simply Proposition 2.1.10: every facet defining inequality of GTSP($G^{\mathrm{s}}_{\mathcal{C}}$) becomes a facet-defining inequality of GRP($\mathit{\Gamma}, \infty$) by 0-node lifting. The second reason is an elegant result of Letchford [Let99], who showed that for every $\mathit{\Gamma}$ there exists a graph $G_{\mathit{\Gamma}}$ such that GRP($\mathit{\Gamma}, \infty$) is a face of GTSP($G_{\mathit{\Gamma}}$), which again, of course, is a face of GTSP($n_{\mathit{\Gamma}}$), where $n_{\mathit{\Gamma}} := |V(G)|$. This implies that every face of GRP($\mathit{\Gamma}, \infty$) can be derived from a facet of GTSP($G_{\mathit{\Gamma}}$) and hence of GTSP($n_{\mathit{\Gamma}}$). Using this, Letchford transferred a number of known facet-defining inequalities for the Symmetric Traveling Salesman Polytope STSP($n_{\mathit{\Gamma}}$) to the GRP.

Hence we see that the GTSP and GRP polyhedra are very closely related, way beyond the fact that the former is a special case of the latter. However, the structure of GRP polyhedra is in general much more complex than that of GTSP polyhedra. Hence, the third reason for focusing on the GTSP polyhedron is that it is possible to address some polyhedral issues more easily in the context of GTSP(n), and obtain results which would be overly complex in the general case.

In this chapter we will first repeat Letchford's result. Then we will address issues of the Graphical Traveling Salesman Polyhedron, focusing on vertices, edges and other faces, then on the relationship of GTSP(n) with the much better studied Symmetric Traveling Salesman Polytope.

5.1 How GRP is a face of GTSP

A facet of GTSP(G) defined by a connectivity inequality for a single node, i.e. $x(\partial(u)) \geq 2$ with $u \in V_n$, is called a *degree facet*. Recall that the inequality defines a facet if and only if u is not a cut-node.

degree facet

In [Let99], a method is proposed to identify GRP($\mathit{\Gamma}, \infty$) with a face of GTSP($n_{\mathit{\Gamma}}$). We repeat the construction here. Let E_R be a multi-set of required edges for $\mathit{\Gamma} = (G, \mathcal{C}, \mathbf{t})$, i.e., $E_R \subseteq E_{\mathcal{C}}$ and $\mathbf{t}(u)$ is equal modulo 2 to the number of edges in E_R incident to u, for all nodes $u \in V(G)$. We construct a graph $G_{\mathit{\Gamma}}$ in the following manner. For each element $uv \in E_R$, add to G a uv-path of length 3, denoting the two new nodes by $w_1(uv)$ and $w_2(uv)$. Using this construction, Letchford obtained the following result.

5.1.1 Theorem ([Let99]) *Let $V := V(G) \subseteq V(G_{\mathit{\Gamma}})$ and $E := E(G) \subseteq E(G_{\mathit{\Gamma}})$. Let F denote the face of GTSP($G_{\mathit{\Gamma}}$) defined by the intersection of the degree facets for all nodes in $E(G_{\mathit{\Gamma}}) \setminus E$*

and the connectivity facets defined by $x(\partial(w_1(uv), w_2(uv))) \geq 2$ *for* $uv \in E_R$. *Then the mapping* $x \mapsto x_{|E}$ *is an affine isomorphism from* F *to* $\mathrm{GRP}(\Gamma, \infty)$.

A consequence of this fact is that the relationship between the GRP polyhedron and the GTSP polyhedron is very close. While it is both obvious and well known (see Section 2.1.1 or [CS98] resp.) that every GTSP-facet can be seen as a facet of $\mathrm{GRP}(\Gamma, \infty)$ for appropriate Γ, Theorem 5.1.1 shows that the reverse holds in a very general manner: every facet of $\mathrm{GRP}(\Gamma, \infty)$ can be found among the facets of GTSP(n) for appropriately chosen n. A conjecture implicit in the work of Naddef & Rinaldi [NR93] said that every facet of GTSP(n) arises from a facet of STSP(n), which would have meant that to know GTSP(n) (and hence, in a way, $\mathrm{GRP}(\Gamma, \infty)$), it would be enough to study the STSP polytope, which has been done extensively. As a matter of fact, all facet-defining inequalities listed in Section 2.1.2 can be derived from well-known STSP facets: odd-cuts from 2-matching, KCs and path-bridges from path inequalities. Many honeycomb inequalities can be derived from binested inequalities.

In the remainder of this chapter, we will dig into the polyhedra GTSP(n) and the relationship between STSP(n) and GTSP(n).

5.2 Notation, terminology, and known facts for GTSP polyhedra

On page vi, a picture of GTSP(3) is shown. The polyhedron is unbounded: all six facets and the three pairs of parallel rays extend infinitely. The picture was created with the help of Maple, and then "artistically improved." GTSP(3) has four vertices, the one in the middle is the Hamiltonian cycle $(1,1,1)^\top$, the three others are permutations of $(2,2,0)^\top$. The three non-negativity facets are the triangle-shaped surfaces, the remaining three facets are degree facets. The intersection of the degree facets is the unique Hamiltonian cycle.

The definitions and facts in this section are elementary. They are extensions of ideas which can be found in [NR93], some of them implicitly. Many of them are also geometrically intuitive, and we encourage the reader to check whether examples for them can be found in the picture on page vi.

shortcut $\mathfrak{s}_{u,e}$
triangle slacks $\bar{t}$
metric vector
TT-set $\Delta_u(a)$

5.2.1 Definition For a node $u \in V_n$ and an edge $e = vw \in E_n$ which forms a triangle with u, i.e., $u \notin e$ holds, we define the *shortcut* $\mathfrak{s}_{u,e} := \chi^e - \chi^{uv} - \chi^{uw} \in \{0, \pm 1\}^{E_n}$. Let $a \in \mathbb{R}^{E_n}$. We define the vector $\bar{t}(a) := (\bar{t}_{u,e}(a))_{u \notin e}$ of *triangle slacks* by $\bar{t}_{u,e}(a) := -\mathfrak{s}_{u,e}\, a$. We will also use the vectors $\bar{t}_u(a)\colon \{e \mid u \notin e\} \to \mathbb{R}$, for $u \in V_n$. A vector $a \in \mathbb{R}^{E_n}$ is called *metric,* if it satisfies the triangle inequality, i.e., if $\bar{t}(a) \geq 0$. For all $u \in V_n$ we define the *TT-set* by

$$\Delta_u(a) := \{ e \in E_n \mid u \notin e \wedge \bar{t}_{u,e}(a) = 0 \}.$$

feasible shortcut
$\Delta_u(F)$

If F is a face of GTSP(n), we say that a shortcut $\mathfrak{s}_{u,e}$ is *feasible for F,* if it is contained in the linear space defined by F, i.e., if $\mathfrak{s}_{u,e} \in \mathrm{lin}(F - F)$. For $u \in V_n$, we denote by $\Delta_u(F)$ the set of all edges forming a triangle with u such that the corresponding shortcut is feasible for u, i.e., $\Delta_u(F)$ is the set of all $e \not\ni u$ for which $\mathfrak{s}_{u,e} \in \mathrm{lin}(F - F)$.

See Fig. 5.1 for a shortcut which can be written as the difference between the two simplest kinds of vertices of GTSP(n), namely the Hamiltonian cycles and the so-called 1-cacti. We will show later (see Lemma 5.3.8) that every feasible shortcut can be written in this way.

Note that the mapping $\bar{t}\colon a \mapsto \bar{t}(a)$ is linear. The following lemma justifies the last part of the definition.

5.2.2 Lemma *Let* F *be a face of GTSP(n) and let* (a, α) *be a valid inequality defining* F. *Then* $\Delta_u(F) \subseteq \Delta_u(a)$ *for all* u. *If* F *is not contained in a non-negativity facet, then equality holds, and the TT-sets do not depend on the inequality defining* F.

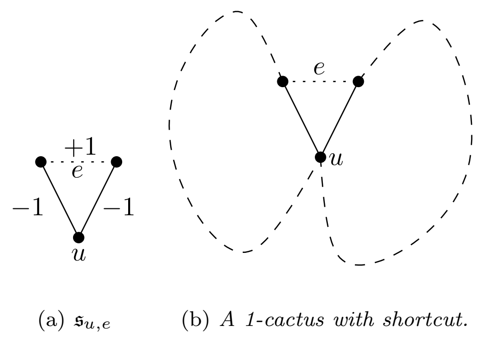


Figure 5.1: A shortcut arising from the difference of a 1-cactus and a Hamiltonian cycle.

Proof. Let $e \in \Delta_u(F)$. Clearly $\mathfrak{s}_{u,e} \in \text{lin}(F - F)$ implies $a\,\mathfrak{s}_{u,e} = 0$, i.e., $e \in \Delta_u(a)$. On the other hand, let $e \in \Delta_u(a)$, and suppose that there exists an $x \in F \cap \mathbb{Z}^{E_n}$ with $x_e \geq 1$. Then $y := x - \mathfrak{s}_{u,e} \in \text{GTSP}(n)$ and $ay = \alpha$, and hence $\mathfrak{s}_{u,e} = x - y \in \text{lin}(F - F)$. □

5.2.3 Lemma *If F, G are faces of GTSP(n) such that $F \cap G$ is not contained in a non-negativity facet, then $\Delta_u(F \cap G) = \Delta_u(F) \cap \Delta_u(G)$.*

Proof. If F is defined by (a, α) and G by (b, β) then $F \cap G$ is defined by $(a + b, \alpha + \beta)$. The result follows from $\bar{t}(a), \bar{t}(b) \geq 0$, the linearity of $\bar{t}$ and the previous lemma. □

5.2.4 Definition A vector $a \in \mathbb{R}^{E_n}$ is said to be *in TT-form,* if it is metric and $\Delta_u(a) \neq \emptyset$ for all $u \in V_n$. *TT-form*

5.2.5 Definition A face F of GTSP(n) is called *metric,* if for every $x \in F$ and every shortcut $\mathfrak{s}_{u,e}$ with $x + \mathfrak{s}_{u,e} \in \text{GTSP}(n)$ we have $e \in \Delta_u(F)$. In other words, F is closed to taking shortcuts in GTSP(n). *metric face*

Examples of metric faces include faces which can be defined by metric inequalities. The following lemma is an extension of a result in [NR93].

5.2.6 Lemma ([NR93]) *Let F be a face of GTSP(n) defined by the inequality (a, α). If F is not contained in a non-negativity facet, then a is metric, which implies that F is metric.*

If F is metric and $\Delta_u(F) = \emptyset$ holds for a node u, then F is contained in the degree facet for the node u. □

We come to the central definition of this chapter. We note that, for any face F of GTSP(n), if F is contained in the degree facet for a node u, then $\Delta_u(F) = \emptyset$.

5.2.7 Definition Let F be a proper face of GTSP(n). We say that F is of *TT-type,* if it is not contained in a non-negativity or a degree facet. *TT-type*

5.2.8 Remark We summarize that a face which is not contained in a non-negativity facet is of TT-type if and only if $\Delta_u(F) \neq \emptyset$ for all u.

Thus, if F is not contained in a non-negativity facet, and F is defined by (a, α), then F is a TT-type face if and only if a is in TT-form. We can now repeat a theorem of Naddef & Rinaldi [NR93] which follows easily from these facts. Fig. 5.2 shows the relationships between the properties of faces of GTSP(n) which we have defined.

5.2.9 Theorem ([NR93]) *The facets of GTSP(n) fall into precisely three types: non-negativity facets, degree facets, and facets defined by TT-form inequalities.*

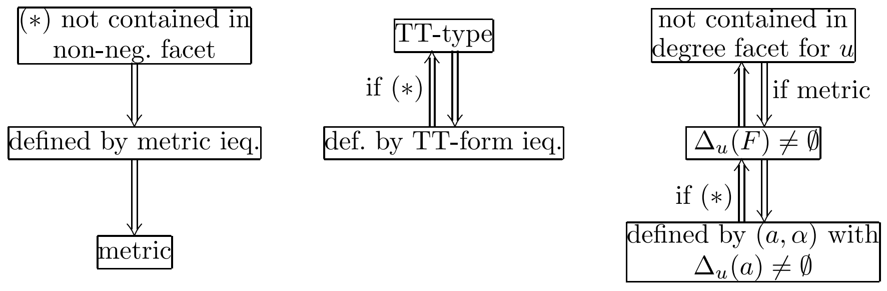

(∗) not contained in non-neg. facet
defined by metric ieq.
metric
TT-type
if (∗)
def. by TT-form ieq.
not contained in degree facet for u
if metric
$\Delta_u(F) \neq \emptyset$
if (∗)
defined by (a, α) with $\Delta_u(a) \neq \emptyset$

Figure 5.2: Relationships between face properties

5.2.1 The Graphical Relaxation

Clearly, the intersection of all degree facets of GTSP(n) is STSP(n). Thus, for every facet H of STSP(n), there exists a facet G of GTSP(n) with $H = G \cap \text{STSP}(n)$. In [NR93], it is shown that the relationship between STSP(n) and GTSP(n) is much closer than what could be expected from this trivial fact. The findings in [NR93] even lead to the conjecture[1] that every TT-type facet G of GTSP(n) induces, by intersection, a facet H of STSP(n). We call a TT-type facet G of GTSP(n) with the property that $G \cap \text{STSP}(n)$ is a facet of STSP(n), a *Naddef-Rinaldi* facet or *NR-facet* for short. A TT-type facet which is not an NR-facet is said to be *non-NR* or a *non-NR facet.* We also speak of the *NR property* or *non-NR property* of a TT-type facet. The above mentioned conjecture thus reads: *there are no non-NR facets;* or: *every TT-type facet is Naddef-Rinaldi.* For ease of reference, we call it the "GR-conjecture", where GR stands for *Graphical Relaxation,* which is the term used for the relationship between the STSP polytope and the GTSP polyhedron.

NR-facet
non-NR
Graphical Relaxation

It is known from [NR93] that if $H := G \cap \text{STSP}(n)$ is a facet of STSP(n) then the facet G is uniquely determined by H—thus the GR-conjecture would imply a one-to-one correspondence between the facets of STSP(n) and those of GTSP(n). Taking into account the introductory remarks at the beginning of this chapter, and the fact that the polytope STSP(n) is certainly one of the best studied polytopes in the area of combinatorial optimization, this conjecture is of great importance for the study of General Routing Polyhedra. Unfortunately, the conjecture turned out to be false (see Section 5.4 below).

From a merely polyhedral point of view, the GR-conjecture was fairly daring anyway. Given a "random" polyhedron of dimension m and a face of dimension $m - n$, few would be inclined to guess that the two polyhedra have essentially the same set of facets. However, despite the intense research on the STSP and GTSP polyhedra, the apparent[2] relevance of the problem, and the fact that it has been open some 15 years, no counterexample has ever been encountered. Moreover, in [ORT05], with computer aid, we were able to prove that the GR-conjecture is true for $n \leq 8$. Finally, it turns out that among the 42,105,161 TT-type facets of GTSP(9), there are only $9! = 362880$ (all equal modulo permutation of nodes), which are non-NR, a surprisingly small number. (We were able to prove the complete outer description of GTSP(9), see Section 5.7, with computer aid and relying heavily on the tilting complexes which we introduce in Section 5.5.) For $n = 10$, we found 288 non-NR facets which are distinct modulo permutation of nodes, in comparison to a (conjectured, see [AP01, CR01]) total number of 15,379 modulo permutation distinct NR-facets. The common sense conclusion to these observations is that the relationship between STSP(n) and GTSP(n) must be exceptionally close, notwithstanding the fact that the GR-conjecture, in its "naive" form, is false. As a central contribution of this chapter we establish this close connection in the form of the theory of tilting complexes.

In the remainder of this section, we review the known facts about the Graphical Relaxation,

[1]We find it necessary to make clear that Naddef & Rinaldi [NR93] as well as Naddef in [Nad02] call this conjecture only a *question,* while it was called a "conjecture implicit in the work of Naddef & Rinaldi" by Applegate et al. [ABCC98, ABCC01]. Carr [Car04] stated it as a conjecture.

[2]The relevance is actually even more apparent in the context of polyhedral combinatorics for the *Symmetric TSP.*

which are prerequisites for understanding the remainder of this chapter, and which lead to the establishment of the GR-conjecture.

5.2.10 Definition ([NR93]) We define a mapping $\bar\lambda\colon \mathbb{R}^{E_n} \to \mathbb{R}^{V_n}$ by letting

$\bar\lambda$

$$\bar\lambda_u(a) := \min_{e \not\ni u} \bar t_{u,e}(a) \tag{5.1}$$

for $a \in \mathbb{R}^{E_n}$ and all $u \in V_n$. Further, we abbreviate

$$\delta_u := \tfrac{1}{2}\chi^{\partial(u)} \tag{5.2}$$

δ_u

and define the following matrix

$$D := \begin{pmatrix} \delta_0 & \dots & \delta_{n-1} \end{pmatrix} \quad \in \mathbb{M}(E_n \times V_n). \tag{5.3}$$

matrix D

It is a long-known fact (see e.g. [GP79]) that $\operatorname{rk} D = n$, and that $\dim \mathrm{STSP}(n) = m - n$, whence the *degree equations* $x(\partial(u)) = 2$, or in our notation $\delta_u x = 1$, for $u \in V_n$, form a complete system of equations for STSP(n). The next lemma is implicit in [NR93]. We will need the second part of it in Section 5.5.

degree equations

5.2.11 Lemma *For every $a \in \mathbb{R}^{E_n}$ there exists precisely one TT-form vector in the set $a + \operatorname{Im} D$, namely $a - D\,\bar\lambda(a)$, and if $a - D\xi$ is in TT-form, then $\xi = \bar\lambda$.*

Proof. Let $a' := a - D\,\bar\lambda(a)$. The following simple calculation shows that $\bar t(a') \geq 0$. For $v \in V_n$ we have

$$\bar t_v(a - D\,\bar\lambda(a)) = \bar t_v(a) - \sum_u \bar\lambda_u \bar t_v(\delta_u) = \bar t_v(a) - \bar\lambda_v(a)\bar t_v(\delta_v) = \bar t_v(a) - \bar\lambda_v(a)\cdot\mathbf{1} \geq 0.$$

In the last inequality, equality holds for the edges e in which the minimum in (5.1) is attained, hence, $\Delta_v(a) \neq \emptyset$. We have proven that a' is in TT-form. For the second statement, it suffices to note that D has full column-rank. □

The vector $a - D\,\bar\lambda(a)$ is sometimes called the *TT-form representative* of a. The next theorem is one of the cornerstones of the Graphical Relaxation.

TT-form representative

5.2.12 Theorem ([NR93]) *Let H be a facet of STSP(n). The facet G of GTSP(n) with the property that $G \cap$ STSP(n) $= H$ is uniquely determined. If G is not a non-negativity facet and H is defined by the inequality (a, α), then G is defined by the following inequality*

$$(a, \alpha) - \left(\begin{smallmatrix} D \\ \mathbf{1}^\top \end{smallmatrix}\right)\bar\lambda(a).$$

For the fact that a TT-form inequality (a, α), which defines a facet H of STSP(n) which is not a non-negativity facet, defines a facet of GTSP(n) (which is implicit in our formulation of the theorem), the paper [NR93] gives a proof similar to the one we use for Proposition 5.3.15 below, but a much simpler argument can be used: There exists a facet G of GTSP(n) which contains H, it is defined by a TT-form inequality by Theorem 5.2.9, and since the TT-form is unique by Lemma 5.2.11, it must be equal to (a, α) except for a strictly positive scalar multiple. The following corollary will be helpful in later sections. We denote the blocking polyhedron for GTSP(n) by $B(n) := B(\mathrm{GTSP}(n))$.

$B(n)$

5.2.13 Corollary *Let $(a, 1)$ define an NR-facet of GTSP(n). Then the face $a \vee \bigvee_{u \in V_n} \delta_u$ of $B(n)$ is an n-simplex.*

The corollary of the last proposition of this section implies that the intersection of a TT-type face of GTSP(n) with STSP(n) is never contained in a non-negativity facet, which is the starting point of our results on the Graphical Relaxation.

$\mathscr{H}(n)$ $|\cdot|_1$ We denote by $\mathscr{H}(n)$ the set of all incidence vectors of (edge sets of) Hamiltonian cycles of the complete graph K_n, and for $x \in \mathbb{R}_+^{E_n}$ we let $|x|_1$ stand for the value $\sum_e x_e$. We note that in [NR93], the following proposition and corollary are given only for facets, but we find it insightful to look at lower dimensional faces, too.

5.2.14 Proposition ([NR93]) *Let F be a metric face of GTSP(n). Let $x \in F \cap \mathbb{Z}^{E_n}$, $u \in V_n$ with $x(\partial(u)) \geq 4$, and $f \in E_n$ with $x_f \geq 1$. Then there exists a shortcut $\mathfrak{s}_{u,e}$ which is feasible for F, i.e., such that $y := x + \mathfrak{s}_{u,e} \in F$, and $y_f \geq 1$. Note also that the length of the vector decreases: $|x|_1 = |y|_1 + 1$.* □

5.2.15 Corollary ([NR93]) *Let F be a face of GTSP(n) not contained in a non-negativity inequality. For every $e \in E_n$, there exists an $x \in \mathscr{H}(n) \cap F$ with $x_e = 1$. In other words, $F \cap STSP(n)$ is not contained in a non-negativity facet.* □

5.3 Vertices, shortcuts, and faces

This section contains preliminary results which are of fairly minor interest in themselves. Thus we feel obliged to give some motivation to the reader to commit himself to it. This section deals with vertices mainly. The set of vertices of GTSP(n) (or even GRP(Γ, ∞)) has, to our knowledge, not been studied at all. It is not our intention to indulge in this line of research more than necessary. In 5.3.1, we give some basic facts about vertices. We name the most intuitive class of vertices, which correspond to polygonal cacti with bridges. In 5.3.2 we show that shortcuts are edges of GTSP(n) and propose an inductive algorithm to enumerate all vertices of GTSP(n). We note that not even the complexity of the recognition problem for GTSP vertices is known, i.e., the question of deciding whether a given vector $x \in \mathbb{Z}_+^{E_n}$ is a vertex of GTSP(n). Obviously, invoking Caratheodory's theorem, the problem is in co**NP**, but nothing is known beyond that. In 5.3.3, we give some facts about the relationship between vertices of GTSP(n) and shortcuts. It might be worthwhile to read this subsection, because it helps in understanding the theory of tilting complexes in the Sections 5.5, 5.8, and the results on 0-node lifting in Section 5.9. We also obtain a result which states that, for many purposes, it is enough to look at a small subset of the vertices, namely the Hamiltonian cycles and so-called 1-cacti, which are called "almost Hamiltonian cycles" in [NR93]. For computational purposes it is particularly important to reduce the number of vertices. For example the fact that two TT-type facets are adjacent on GTSP(n) if they are adjacent on the polytope defined by the Hamiltonian cycles and 1-cacti has proven very useful: this special case was used for [ORT05] and the computational part of 5.7.3. Finally, in 5.3.4, we turn to the question of adjacency of facets on the GTSP polyhedron, which, being a more intuitive, geometric, and important topic, does not require a vindication.

5.3.1 Vertices

In this section we deal with the vertices of GTSP(n).

$\mathscr{E}(n)$ **5.3.1 Definition** We denote by $\mathscr{E}(n)$ the set of vertices of GTSP(n). Further, for $l \geq 0$, we let

$$\mathscr{E}^l(n) := \{x \in \mathscr{E}(n) \mid |x|_1 = n + l\},$$

and $\mathscr{E}^{\leq l}(n) := \bigcup_{j=0}^{l} \mathscr{E}^j(n)$.

5.3.2 Remark We note some trivial facts about vertices of GTSP(n).

(a). If $x \in \mathscr{E}(n)$, then $x_e = 2$ iff e is a cut-edge of the graph $(V_n, E(x))$.

n	$\lvert\mathscr{C}(n)\rvert$
1	1
2	1
3	4
4	31
5	362
6	5676
7	111982
8	2666392
9	74433564
10	2384579440
11	86248530296
12	3476794472064
13	154579941792256
14	7514932528712896
15	396595845237540600

Table 5.1: Number of cacti [Slo05]

(b). Vertices of GTSP(n_1) and GTSP(n_2) can be glued together by 1-sum operations to form vertices of GTSP($n_1 + n_2 - 1$). There are other "sum"-operations by which vertices can be glued together.

(c). An edge e of the graph $(V_n, E(x))$ for a vertex x with $x_e = 1$ can be subdivided, and the new vector is a vertex of GTSP($n+1$).

(d). Given any Eulerian multigraph, by subdividing each edge with at least two new nodes, taking the incidence vector and filling in zeros, we obtain a GTSP vertex [Let05b].

Now we address the most fundamental type of vertices.

5.3.3 Definition A graph of which every block is an edge or a circle is called a *polygonal cactus with bridges,* or just cactus for short. We identify a cactus C on the node set $V(C) = V_n$ with a vector $z^C := \chi^{E(C)} + \chi^B \in \mathbb{R}^{E_n}$ where B shall denote the set of bridges of C. This means that $z^C_e = 0$ if e is not an edge of the cactus, $z^C_e = 1$ if e is contained in a circle of C and $z^C_e = 2$ if $e \in E(C)$ is not contained in a circle of C. We denote by $\mathscr{C}(n)$ the set of all (vectors of) cacti with node set V_n. For $l \geq 0$ we let $\mathscr{C}^l(n) := \{z^C \in \mathscr{C}(n) \mid \left|z^C\right|_1 = n + l\}$, and we call the elements of this set *l-Cacti.* Further we define $\mathscr{C}^{\leq l}(n) := \bigcup_{j=0}^{l} \mathscr{C}^j(n)$.

cactus

$\mathscr{C}(n)$

l-Cacti

Clearly $\mathscr{C}(n) \subseteq$ GTSP(n). Note that $\mathscr{H}(n) = \mathscr{C}^0(n)$. The set of Naddef & Rinaldi's [NR93] so-called *almost Hamiltonian cycles* is just $\mathscr{C}^1(n)$. The items (b) and (c) of Remark 5.3.2 imply that $\mathscr{C}(n) \subseteq \mathscr{E}(n)$. Moreover, it is easy to see that $\mathscr{E}^0(n) = \mathscr{C}^0(n) = \mathscr{H}(n)$ and $\mathscr{E}^1(n) = \mathscr{C}^1(n)$ holds.

The number of polygonal cacti with bridges on a given number of nodes is known [FU56]. If $c_n := |\mathscr{C}(n)|$, $n \in \mathbb{N}$, denotes the sequence of numbers of cacti then for all n

$$c_n = e^{\frac{2nc_n - (nc_n)^2}{2 - 2nc_n}}.$$

Table 5.1 shows the number of cacti for $n = 1, \dots, 15$. We have already noted that $\mathscr{C}(n)$ is a subset of $\mathscr{E}(n)$. Using computational methods, we were able to determine $|\mathscr{E}(n)|$ for $n = 1, \dots, 7$, and we found that $|\mathscr{E}(n)| = |\mathscr{C}(n)|$ holds in this range. For $n \geq 8$ there are vertices of GTSP(n) which are not cacti. See for example the non-cactus vertex of GTSP(8) displayed in Fig. 5.3, which is determined as a vertex by the non-negativity, connectivity and comb-inequalities which it satisfies with equality. We summarize these results in the following proposition.

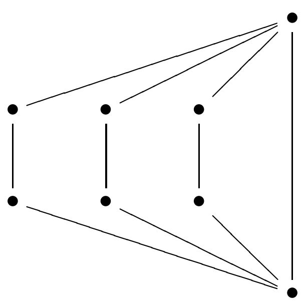


Figure 5.3: Non-cactus vertex of GTSP(8)

5.3.4 Proposition *$\mathscr{E}(n) \supseteq \mathscr{C}(n)$, and equality holds if and only if $n \le 7$.* □

5.3.2 Some edges of GTSP(n)

We now briefly deal with the most obvious edges of the polyhedron GTSP(n).

5.3.5 Proposition *Let $x \in GTSP(n)$ and $u \in V_n$ with $x(\partial(u)) \ge 4$ such that $x + \mathfrak{s}_{u,e} \in GTSP(n)$. If x is a vertex, then $x + \mathfrak{s}_{u,e}$ is a vertex and the line segment $[x, x + \mathfrak{s}_{u,e}]$ is an edge of GTSP(n).*

Proof. Let (A, b) be the system of facet-defining inequalities which are satisfied with equality by x, except for the non-negativity inequality $(\chi^e, 0)$.

By the first part of Lemma 5.2.6, for $t \in \mathbb{R}_+$, the vector $x^t := x + t\mathfrak{s}_{u,e}$ satisfies $Ax^t = b$, hence, since x is a vertex, the line segment $[x, x + \mathfrak{s}_{u,e}]$ must be contained in an edge. We only need to show that x^1 is the other end vertex of the edge. Let us denote by v, v' the end nodes of e. If $x_{uv} = 1$ or $x_{uv'} = 1$, then $x^t \notin \mathbb{R}_+^{E_n} \supseteq$ GTSP(n) for $t > 1$, and thus x^1 is a vertex. If $x_{uv}, x_{uv'} \ge 2$, then, since x is a vertex, we have $x_{uv} = x_{uv'} = 2$, and, by Remark 5.3.2-(a), the graph $(V_n, E(x) \setminus \{uv, uv'\})$ is not connected. Let U denote the node set of the connected component of this graph containing u. Then x^1 satisfies the connectivity inequality $x(\partial(U)) \ge 2$ with equality, and x^t violates it for $t > 1$. Hence, x^1 is the other-end vertex of the edge. □

It can be seen from Table 5.1 that the number of vertices of GTSP(n) is quite high. To our knowledge, no practicable is known to enumerate a superset of the vertices in reasonable time. We propose a method based on Proposition 5.3.5, which involves the solution of large LP-feasibility problems, as displayed in Algorithm 5.3.2. The proof that the algorithm produces all and only vertices of GTSP(n) can easily be done. Given that the combinatorial knowledge about the set of vertices of GTSP(n) is virtually nonexistent, a practicable algorithm is very likely to rely on LP-feasibility, but it might adapt step 12 of Algorithm 5.3.2 to reduce the number of LP-feasibility problems as far as possible, for example by recognizing all vectors which are composed from vertices of GTSP(k), $k < n$, by, e.g., 1-sum operations (they are vertices).

By adapting step 1 and restricting the set of shortcuts considered in steps 4–10, Algorithm 5.3.2 can also be used to enumerate all vertices on a face defined by a given metric inequality. We used it this way for the computational part of the proof of a complete description of GTSP(9), see Section 5.7.

5.3.3 Relationship between vertices and faces

5.3.6 Lemma *Let F be a face of GTSP(n) defined by a metric inequality (a, α). Then F is uniquely determined by the sets $\mathscr{H}(n) \cap F$ and $\Delta_u(a)$, $u \in V_n$. In other words, if (a', α') is another metric valid inequality such that $\Delta_u(a) = \Delta_u(a')$ for all u, and the face F' defined by (a', α') satisfies $F' \cap \mathscr{H}(n) = F \cap \mathscr{H}(n)$, then $F = F'$.*

Algorithm 5.1 Iterative vertex enumeration for GTSP

```
Input: $n \geq 3$
Output: List of all vertices of GTSP($n$)
1: Enumerate the set $\mathscr{E}^0(n)$ of all Hamiltonian cycles of the complete graph $K_n$.
2: For $l = 0, \dots$ do
3:   $L := \emptyset$
4:   For $x \in \mathscr{E}^l(n)$ do
5:     For $u \in V_n$ and $e \in E_n$ with $u \notin e$ do
6:       If $x - \mathfrak{s}_{u,e} \in$ GTSP($n$) then
7:         $L := L \cup \{x - \mathfrak{s}_{u,e}\}$
8:       End if
9:     End for
10:  End for
11:  For $y \in L$ do
12:    If $y$ can be written in the form $\sum_i \xi_i x_i + \zeta_e \chi^e$, where $\xi, \zeta \geq 0$, $\mathbf{1}\xi = 1$, and
       $x_i \in \mathscr{E}^l(n) \cup L \setminus \{y\}$ then
13:      Remove $y$ from $L$
14:    End if
15:  End for
16:  If $L = \emptyset$ then
17:    Stop: all vertices have been constructed.
18:  End if
19:  $\mathscr{E}^{l+1}(n) = L$.
20: End for
```

Proof. Every vertex of F is either in $\mathscr{H}(n)$, or it can be constructed from a Hamiltonian cycle by successively subtracting shortcuts in $\Delta_u(a)$, $u \in V_n$. Further, $\mathbb{R}_+ \cdot \chi^{uv}$ is a ray of F iff $a_{uv} = 0$ iff $vw \in \Delta_u(a) \wedge uw \in \Delta_v(a)$ for $w \notin \{u, v\}$. $\square$

5.3.7 Remark If $x \in \mathscr{E}(n)$, and $\mathfrak{s}_{u,vw}$ is a shortcut, then $x + \mathfrak{s}_{u,vw} \notin$ GTSP(n) is caused by one of the following obstructions:

(a). $x_{uv} = 0$ or $x_{uw} = 0$.

(b). We have $x_{uv} = x_{uw} = 1$, and there exists a block of the graph $(V_n, E(x))$ which contains u, v, and w, and such that v and w are the only neighbors of u in that block.

5.3.8 Lemma *Let F be a metric face of GTSP(n). The set of feasible shortcuts of F is determined solely by the Hamiltonian cycles and the 1-cacti in F. To be precise, if we let $F^0 := F \cap \mathscr{H}(n)$ and $F^1 := F \cap \mathscr{C}^1(n)$, then, for all u,*

$$\Delta_u(F) = \{e \mid u \notin e \ \wedge \ \mathfrak{s}_{u,e} \in F^0 - F^1\}.$$

Proof. Let $\mathfrak{s}_{u,e}$ be a feasible shortcut for F, where $e =: v'v$. Then there exist $z \in F \cap \mathbb{Z}^m$ with $z_e \geq 1$, and we have $z' := z - \mathfrak{s}_{u,e} \in F \cap \mathbb{Z}^m$. Now let $z = y_0, \dots, y_r$ be a sequence of maximal length consisting of vectors in GTSP(n) such that for $j = 0, \dots, r-1$, the difference $y_j - y_{j+1}$ is a shortcut and $y_j + \mathfrak{s}_{u,e} \in$ GTSP(n) holds for all $j = 0, \dots, r$. We claim that $y := y_r$ is a 1-cactus. If this claim is true, then, since $x := y + \mathfrak{s}_{u,e} \in$ GTSP(n), x must be a Hamiltonian cycle and we have $\mathfrak{s}_{u,e} = x - y \in F^0 - F^1$.

To prove the claim, we show that, if y is not a 1-cactus, then there exists a shortcut $\mathfrak{s}_{w,f}$ with $y' := y + \mathfrak{s}_{w,f} \in$ GTSP(n) and $y' + \mathfrak{s}_{u,e} \in$ GTSP(n). Suppose that y is not a 1-cactus, i.e., we have to distinguish two cases:

1. there exists a node $w \neq u$ with $y(\partial(w)) \geq 4$, or

2. $y(\partial(u)) \geq 6$.

We make use of the remark above. Let C denote the block of the graph $(V_n, E(y))$ which contains u and v. If $v' \notin C$, then the existence of a y' is easy to see. Thus, we assume $v' \in C$. Since $y + \mathfrak{s}_{u,e} \in \text{GTSP}(n)$, we have $y(\partial(C \setminus \{u\})) \geq 4$. In the first case, we have two sub-cases, namely $w \in e$ or $w \notin e$. If $w \in e$, assuming w.l.o.g. that $w = v$, if there exists an edge $ww' \in E(C)$ with $y_{ww'} \neq 0$, then a shortcut $\mathfrak{s}_{w,f}$ can easily be found. Otherwise we note that $\delta := y(\partial(C) \setminus \partial(u)) \geq 3$. Any shortcut $\mathfrak{s}_{w,f}$ can reduce δ by at most two, so any $\mathfrak{s}_{w,f}$ which does not use the edge uv has the desired property. The sub-case $w \notin e$ is easy, as well as the second major case. □

When looking at an affine space which is the affine hull of some set of vertices of GTSP(n) possibly plus rays $\mathbb{R}_+ \cdot \chi^e$, we will want to distinguish between the Hamiltonian cycles, and the "rest", i.e., the vertices "modulo" the Hamiltonian cycles. To be precise, if A is such an affine space and $A_{\mathscr{H}} := \operatorname{aff}(A \cap \mathscr{H}(n))$, let us denote by $L := A - A$ the linear space defined by A, and by $L_{\mathscr{H}} := A_{\mathscr{H}} - A_{\mathscr{H}} = L \cap \ker D^\top$ (the matrix D is defined in (5.3) on page 53) the linear subspace of L corresponding to the Hamiltonian cycles. Clearly A and L are isomorphic as affine spaces, the isomorphism takes $A_{\mathscr{H}}$ onto $L_{\mathscr{H}}$, and the intuitive notion "A modulo $A_{\mathscr{H}}$" corresponds to $L/L_{\mathscr{H}}$. The next lemma will make clear that in the case that $A = \operatorname{aff} F$ for a metric face F, the "vertices modulo the Hamiltonian cycles" are just the shortcuts viewed modulo $L_{\mathscr{H}}$. This point of view will be useful later.

5.3.9 Lemma *Let F be a metric face of GTSP(n). The space L is generated by members of $L_{\mathscr{H}}$ and by shortcuts. In other words, the linear space $L/L_{\mathscr{H}}$ is generated by the set of (projected images of) feasible shortcuts of F. To be precise, if $\pi\colon L \to L/L_{\mathscr{H}}$ is the canonical projection, then $L/L_{\mathscr{H}}$ is generated by*

$$\big\{\pi(\,\mathfrak{s}_{u,e}\,) \;\big|\; e \in \Delta_u F\big\}.$$

Proof. Let $z, z' \in F$. By the previous lemma, there exist Hamiltonian cycles x and x' as well as shortcuts s_i, $i = 0, \dots, k$, and s'_j, $j = 0, \dots, l$ with $z = x + \sum s_i$ and $z' = x' + \sum s'_j$. Hence $z' - z$ is in the space generated by elements of $L_{\mathscr{H}}$ and shortcuts. □

As a result we obtain the following proposition, which is an extension of a result in [ORT05].

5.3.10 Proposition *Let F be a metric face of GTSP(n). Then we have*

$$\dim F = \dim \mathscr{C}^{\leq 1}(n) \cap F,$$

or, in other words, the Hamiltonian cycles and the 1-cacti suffice to determine the dimension of a face of GTSP(n) which can be defined by a metric inequality.

Proof. Let $A := \operatorname{aff} F$, $L := A - A$, and $A^1 := \operatorname{aff}(F \cap \mathscr{C}^{\leq 1}(n))$, $L^1 := A^1 - A^1$. By the previous lemmas, we have $L/L_{\mathscr{H}} = L^1/L_{\mathscr{H}}$, and hence

$$\dim F = \dim L = \dim L_{\mathscr{H}} + \dim L/L_{\mathscr{H}} = \dim L_{\mathscr{H}} + \dim L^1/L_{\mathscr{H}} = \dim L^1 = \dim \mathscr{C}^{\leq 1}(n) \cap F,$$

which proves the proposition. □

In this context, we also note the following useful fact, which is implicit in [NR93].

5.3.11 Lemma *Let $s_u = \mathfrak{s}_{u,e_u}$, $u \in V_n$ be a family of shortcuts. The s_u are linearly independent modulo* $\ker D^\top$.

Proof. Let $\sum_u \xi_u s_u \in \ker D^\top$. For any $v \in V_n$, by applying $\cdot(\partial(v))$, this implies that $-2\xi_v = 0$, whence $\xi = 0$. Thus the s_u are linearly independent modulo $\ker D^\top$. □

5.3.4 Adjacency of facets

We recall that two facets F, G of a polyhedron P are said to be *adjacent,* if their intersection is a ridge of the polyhedron, and that a ridge is a face with codimension two. As a first statement on adjacency of facets of the GTSP polyhedron, we note that Corollary 5.2.13 implies that any two degree facets are adjacent and that every NR-facet is adjacent to every degree facet.

5.3.12 Definition We say that a set of TT-type facets $\mathcal{H}$ is *TT-disjoint* at a node u, if

TT-disjoint

$$\bigcap\nolimits_{H\in\mathcal{H}} \Delta_u(H) = \emptyset.$$

The facets in $\mathcal{H}$ are called *nowhere TT-disjoint*, if they are not TT-disjoint at any node $u \in V_n$.

5.3.13 Proposition *Let $H_0, \dots, H_k$ be TT-type facets of GTSP(n) such that $\bigcap_j H_j$ is not contained in a non-negativity facet. The facets are nowhere TT-disjoint if and only if $\bigcap_j H_j$ is a TT-type face.*

Proof. This follows immediately from Lemma 5.2.3 and Remark 5.2.8. □

5.3.14 Corollary *If G and H are adjacent TT-type facets of GTSP(n), then they are nowhere TT-disjoint.* □

The following lemma plays a central role for the graphical relaxation. It is a generalization of the Theorem 5.2.12.

5.3.15 Proposition *Let $\mathcal{H}$ be a set of NR-facets of GTSP(n) such that $F :=$ STSP(n) $\cap \bigcap_H H$ is a face of codimension c in STSP(n). Suppose that F is not contained in a non-negativity facet. If the facets in $\mathcal{H}$ are nowhere TT-disjoint, then $\bar{F} := \bigcap_H H$ is a TT-type face of codimension at most c in GTSP(n).*

Proof. For all u, let $e_u \in \bigcap_H \Delta_u(H)$, and abbreviate $s_u := \mathfrak{s}_{u,e_u}$. Denote by $L_{\mathscr{H}} := \text{lin}(F - F) \subseteq \ker D^\top$, $L := \text{lin}(\bar{F} - \bar{F})$ and $L_{V_n} := \text{lin}\{s_u \mid u \in V_n\}$. We have to show that $\dim(L_{V_n} + L_{\mathscr{H}})/L_{\mathscr{H}} \geq n$. We show the stronger statement that the dimension of the vector space $L_{V_n} + \ker D^\top / \ker D^\top$ is at least n, by noting that the s_u are linearly independent modulo $\ker D^\top$ by Lemma 5.3.11. □

Let G and H be NR- or non-negativity facets of GTSP(n). We say that G and H are *adjacent on STSP,* if the facets $G \cap$ STSP(n) and $H \cap$ STSP(n) of STSP(n) are adjacent.

adjacent on STSP

5.3.16 Corollary *Let G, H be NR-facets which are adjacent on STSP. If G and H are nowhere TT-disjoint, then they are adjacent as facets of GTSP(n).* □

The two corollaries of this section give rise to the question of what happens in GTSP(n) "between" two facets which are adjacent on STSP, but which are TT-disjoint at some node (if such a combination exists at all—well, it does). Obviously, there must be something "between" two such facets since they cannot be adjacent. This question will be answered in the next section.

5.4 Existence of non-NR facets with codimension 2 in STSP

We now give a mere proof of existence of a non-NR facet of GTSP(n), under the condition that there exists a ridge of STSP(n) which satisfies certain conditions. An extended abstract of parts of this section appeared in the proceedings of the 11th conference Integer Programming and Combinatorial Optimization, IPCO [ORT05]. We are indebted to Marcus Oswald for contributing to some key results of this section. The extended abstract [ORT05] also included a method to extract an inequality defining a non-NR facet of GTSP(n) out of the inequalities which define facets of STSP(n) containing the ridge. This method is made obsolete by the tilting complexes which we will introduce in the next section.

5.4.1 Remark Let (c,γ) define a TT-type facet F of GTSP(n) which is non-NR. There exist inequalities $(a_1,\alpha_1),\dots,(a_l,\alpha_l)$ defining facets of GTSP(n) and STSP(n) (i.e., NR-facets), and $\mu_1,\dots,\mu_l \geq 0$ such that

$$(c,\gamma) = \sum_{j=1}^{l} \mu_j (a_j,\alpha_j) - \sum_{u=0}^{n-1} \bar\lambda_u \cdot (\delta_u, 1), \tag{5.4}$$

where we abbreviate $\bar\lambda := \bar\lambda\bigl(\sum_{j=1}^{l}\mu_j a_j\bigr)$. In other words, (c,γ) is the TT-form representative of $\sum_{j=1}^{l}\mu_j(a_j,\alpha_j)$. The vector $\bar\lambda$ is non-negative, and there exists u with $\bar\lambda_u > 0$. If $\operatorname{codim} F = 2$, then the (a_j,α_j) are unique up to scaling.

good face

5.4.2 Definition A *good face* of STSP(n) is a proper face which is not contained in a non-negativity facet.

Corollary 5.2.15 shows that for any TT-type facet G of GTSP(n), the intersection $G \cap$ STSP(n) is a good face of STSP(n). We will now take the reverse point of view and consider good faces of STSP(n). Let F be a fixed good face of STSP(n), and let $\{H_0,\dots,H_k\}$ be the set of NR-facets of GTSP(n) containing F. This means that $\{\text{STSP}(n)\cap H_0,\dots,\text{STSP}(n)\cap H_k\}$ is the set of facets of STSP(n) containing F. For $j = 0,\dots,k$, let (a_j,α_j) be an inequality defining H_j. We define the following matrix

matrix A

$$A := \begin{pmatrix} a_0 & \dots & a_k \\ \alpha_0 & \dots & \alpha_k \end{pmatrix} \in \mathbb{M}\Bigl((E_n \times \{\text{rhs}\}) \times (\{0,\dots,k\})\Bigr). \tag{5.5}$$

For $u \in V_n$ and $e \in E_n$ with $u \notin e$ we define

t, $t_{u,e}$

$$t_{u,e}\colon \mathbb{R}^{k+1} \to \mathbb{R}\colon \mu \mapsto \bar t(\operatorname{pr}_{E_n} A\mu) = \sum_{j=0}^{k} \mu_j\, \bar t_{u,e}(a_j) = \Bigl(\bar t_{u,e}(a_0) \quad \dots \quad \bar t_{u,e}(a_k)\Bigr)\mu.$$

tilting function λ

For $u \in V_n$ we define the *tilting functions* λ_u by

$$\lambda_u\colon \mathbb{R}^{k+1} \to \mathbb{R}\colon \mu \mapsto \min_{e \not\ni u} t_{u,e}(\mu),$$

and we let $\lambda := (\lambda_0,\dots,\lambda_{n-1})^\top\colon \mathbb{R}^{k+1} \to \mathbb{R}^{V_n}$. The following fact is an immediate consequence of the definitions. (We recall that a mapping f is called *positive homogeneous,* if $f(\alpha x) = \alpha f(x)$ for all $\alpha \in \mathbb{R}_+$ and all x.)

5.4.3 Lemma *λ is continuous, piecewise linear and positive homogeneous, and the λ_u, $u \in V_n$, are concave.* □

$\mu \mapsto (c_\mu,\gamma_\mu)$

The key idea now is to consider the following mapping

$$\mathbb{R}^{k+1} \to \mathbb{R}^{E_n} \times \mathbb{R}\colon \mu \mapsto (c_\mu,\gamma_\mu) := A\mu - \bigl(\begin{smallmatrix} D \\ \mathbf{1}^\top \end{smallmatrix}\bigr)\lambda(\mu).$$

5.4.4 Lemma *For all $\mu \geq 0$ the inequality (c_μ,γ_μ) is valid for GTSP(n).*

Proof. Since c_μ is metric, the validity of (c_μ,γ_μ) for GTSP(n) follows from its validity for STSP(n), which is given because it is a sum of valid inequalities. □

G_μ

For $\mu \geq 0$, let G_μ denote the face of GTSP(n) defined by (c_μ,γ_μ). The proof of the following lemma is identical to that of Proposition 5.3.15. I am indebted to M. Oswald for contributing to the first version of the proof.

5.4.5 Lemma *The inequality (c_μ,γ_μ) defines a face of codimension at most* $\operatorname{codim} F$ *of GTSP(n). In other words:* $\operatorname{codim}_{\text{GTSP}(n)} G_\mu \leq \operatorname{codim}_{\text{STSP}(n)} F$

Proof. As c_μ is in TT-form, we can choose, for each node u, a shortcut $s_u := \mathfrak{s}_{u,e_u}$ with $e_u \in \Delta_u(c_\mu) = \Delta_u(G_\mu)$. The result then follows from Lemma 5.3.11. □

It is easy to see that $\gamma_\mu \neq 0$ for all μ, see Lemma 5.5.15 below. Thus, we note that $\frac{1}{\gamma_\mu} c_\mu \in B(n)$. To be more precise, $\frac{1}{\gamma_\mu} c_\mu$ is in the (codim $F - 1$)-skeleton of $B(n)$, i.e., the union of all faces of $B(n)$ with dimension at most codim $F - 1$. Here is an only slightly less obvious fact.

5.4.6 Lemma *$\frac{1}{\gamma_\mu} c_\mu$ is a convex combination of vertices of $B(n)$ corresponding to TT-type facets of GTSP(n). It is a relative interior point of a bounded face of $B(n)$.*

Proof. Since F is a good face, no non-negativity summand $\zeta\chi^e$ with $\zeta > 0$ can occur in a representation of $c_\mu/\gamma_\mu \in \operatorname{conv} B + \operatorname{cone}\{\chi^e \mid e \in E_n\}$. No degree vertex δ_u can occur in a convex combination of c_μ/γ_μ, because a sum of TT-form left-hand-sides satisfies the triangle inequality and c_μ/γ_μ is in TT-form.

Note that the fact that the face of $B(n)$ of which $\frac{1}{\gamma_\mu} c_\mu$ is a relative interior point is bounded also follows from (d). □

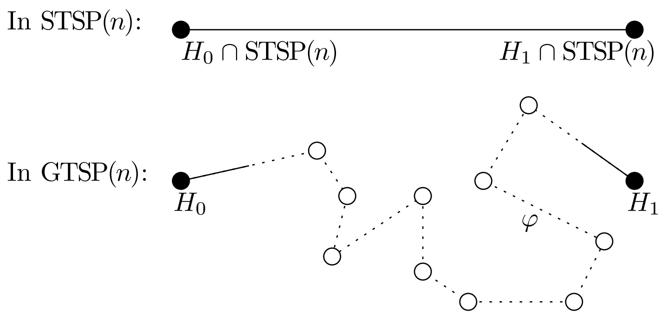


Figure 5.4: Curve on the boundary of B(GTSP(n)) defined by adjacent facets of STSP(n).

Now we restrict ourselves to the case that F is a ridge, i.e., codim $F = 2 = k + 1$. We consider the mapping

$$\varphi\colon [0,1] \mapsto B(n)\colon s \mapsto \frac{1}{\gamma_{(1-s,s)}} c_{(1-s,s)}.$$

It is a continuous curve by Lemma 5.4.3, which starts in the vertex of $B(n)$ corresponding to H_0, walks along the graph of $B(n)$, and ends in the vertex corresponding to H_1. If H_0 and H_1 are TT-disjoint at at least one node, then, by Corollary 5.3.14, H_0 and H_1 are not adjacent, and hence φ must hit another vertex along the way. See Fig. 5.4 for an illustration. This additional vertex corresponds to a TT-type facet, which, clearly, cannot be NR. Hence, to falsify the GR-conjecture, we "only" have to establish two adequate facets H_0 and H_1. For $n = 9$, inequalities defining such facets are displayed in Tab. 5.2.

The process of moving a point continuously along the boundary of $B(n)$ can also be visualized as continuously tilting a valid supporting hyperplane, see Fig. 5.5.

5.5 Tilting complexes

Let F be a proper face of STSP(n) which is not contained in a non-negativity facet. Denote by $\{\bar{H}_0, \dots, \bar{H}_k\}$ the set of all facets of STSP(n) containing F, and by (a_j, α_j) an inequality in TT-form defining $\bar{H}_j$. Let H_j be the NR-facet of GTSP(n) defined by (a_j, α_j). In this section we explain how, using only the information contained in the coefficients and right hand sides of the inequalities (a_j, α_j), we can construct the geometry of all TT-type faces of GTSP(n) which contain F.

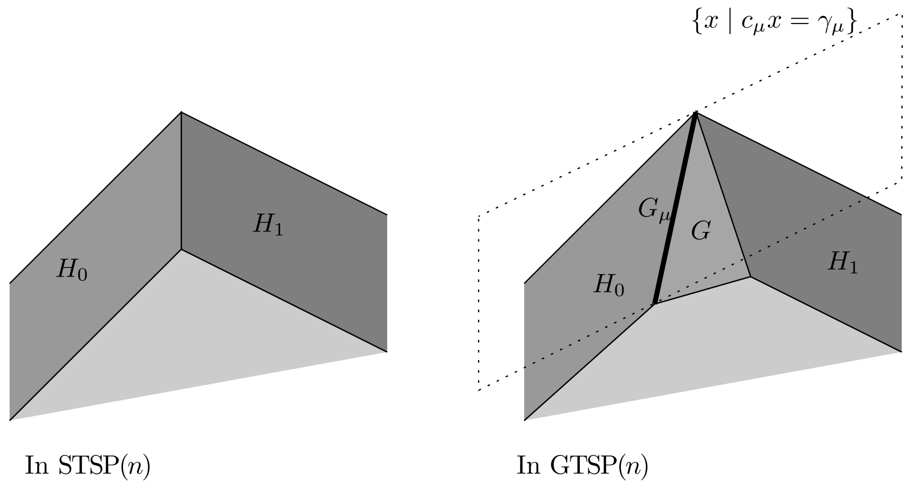
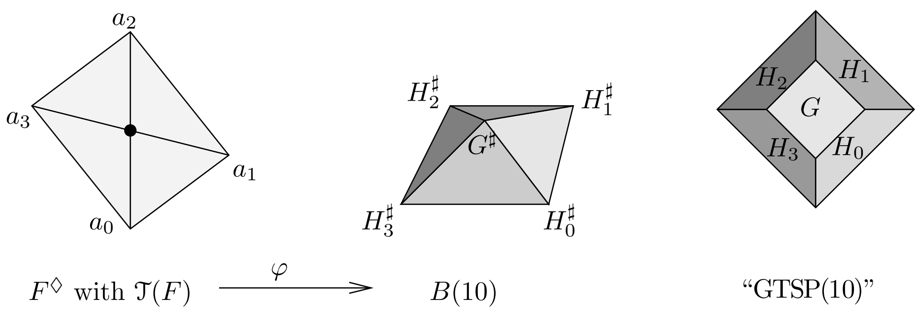
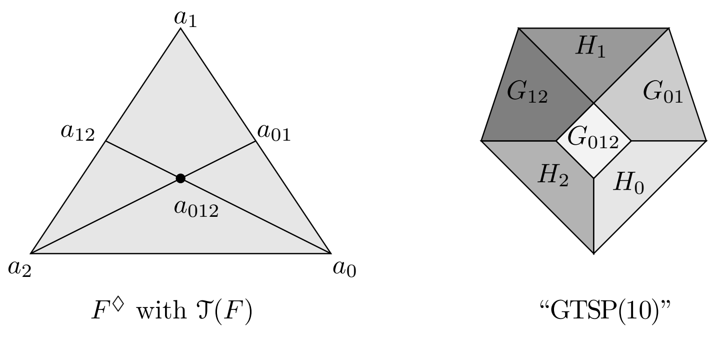


Figure 5.5: Tilting a hyperplane



(a) $k = 3$



(b) $k = 2$

Figure 5.6: Tilting complex for two faces F of STSP(10) with $\operatorname{codim} F = 3$.

	0	1	2	3	4	5	6	7	8
0	0	4	3	3	2	3	5	1	2
1	4	0	1	3	2	5	3	3	2
2	3	1	0	2	3	4	4	2	3
3	3	3	2	0	1	4	4	2	3
4	2	2	3	1	0	3	5	**3**	**4**
5	3	5	4	4	3	0	2	**4**	**5**
6	5	3	4	4	5	2	0	4	3
7	1	3	2	2	**3**	**4**	4	0	1
8	2	2	3	3	**4**	**5**	3	1	0

≥ 18; SMAPO # 47

	0	1	2	3	4	5	6	7	8
0	0	6	4	4	6	4	6	2	3
1	6	0	2	6	4	6	4	4	3
2	4	2	0	4	6	**8**	6	**6**	5
3	4	6	4	0	2	**8**	6	**6**	5
4	6	4	6	2	0	6	8	6	7
5	4	6	**8**	**8**	6	0	2	4	5
6	6	4	6	6	8	2	0	4	3
7	2	4	**6**	**6**	6	4	4	0	1
8	3	3	5	5	7	5	3	1	0

≥ 28; SMAPO # 121

Table 5.2: Two inequalities defining NR-facets of GTSP(9), which are adjacent on STSP and TT-disjoint at 0 (the bold numbers are the tight triangles for node 0).

Our main result is the following. Using only (a_j, α_j), $j = 0, \dots, k$, we can construct a polytopal complex $\mathcal{T}(F)$ which subdivides $F^\diamondsuit$ and a continuous, injective, piecewise projective mapping $\varphi\colon F^\diamondsuit \to \mathbb{R}^{E_n}$, such that $P \mapsto \varphi(P)$ defines a combinatorial isomorphism[3] between the polytopal complex $\mathcal{T}(F)$ and the subcomplex of the boundary complex of $B(\mathrm{GTSP}(n))$, which consists of faces of the blocking polyhedron corresponding to the TT-type faces of GTSP(n) containing F.

We will detail the construction of $\mathcal{T}(F)$ and φ in the remainder of this section. For now, let us look at some examples for STSP(10). Fig. 5.6(a) refers to a face F of STSP(10) with codimension 3 which is the intersection of four facets of STSP(10). On the left, $F^\diamondsuit$ with the tilting complex $\mathcal{T}(F)$ is displayed. The labeling of the vertices of $F^\diamondsuit$ with $a_0, \dots, a_3$ is informal. The middle picture shows the subcomplex of the boundary complex of $B(\mathrm{GTSP}(10))$ which corresponds to the TT-type faces of GTSP(10) containing F. The picture on the right is merely illustrative, since GTSP(10) has dimension 45. It shows the adjacency and intersection relations between the facets of GTSP(10), which really have dimension 44 instead of 2. It can be read from the tilting complex $\mathcal{T}(F)$ that the face F is the intersection of a non-NR facet G of GTSP(10) with STSP(10), and that G is adjacent to each of the four facets $H_0, \dots, H_3$. It can also be seen that H_i is adjacent to H_{i+1} (where the index is taken modulo 4) in GTSP(10), and that $H_i \cap H_{i+1} \cap G$ is a face of codimension 3 of GTSP(n), which is not contained in any other facet of GTSP(10).

Fig. 5.6(b) refers to a face F of STSP(10) with codimension 3 which is the intersection of three facets, H_0, H_1, H_2 of STSP(10). The corresponding face $F^\diamondsuit$ of the polar of STSP(10) is a triangle, which is displayed on the left, together with the tilting complex $\mathcal{T}(F)$. It can be read from $\mathcal{T}(F)$ that there are three non-NR facets of GTSP(10) which contain F. In fact, the intersection of G_{012} with STSP(10) is equal to F, while the intersection of $G_{i\,j}$ with STSP(10) equals $H_i \cap H_j \cap \mathrm{STSP}(10)$. G_{012} is adjacent to H_0, G_{01}, G_{12}, and H_2, but not to H_1. H_0 is adjacent to H_2 and G_{01}. The intersection of H_0 and H_1 is not a TT-type face. The face $G_{012} \cap G_{01} \cap H_1 \cap G_{12}$ of GTSP(n) has codimension 3 and is not contained in any other facet of GTSP(n). Similar statements can be read for the other adjacency and intersection relations. As far as possible by reducing the dimension to 3 of GTSP(n), the relationships are illustrated by the picture on the right.

5.5.1 Some technical prerequisites

In this section, we let x^* be an arbitrary relative interior point of STSP(n). Let D be as in (5.3). We denote by $P\colon \mathbb{R}^{E_n} \to \ker D^\top$ the orthogonal projection. By choosing an orthonormal basis of $\ker D^\top$ with respect to the scalar product inherited from $\mathbb{R}^{E_n}$, we construct an isometric linear isomorphism $\Psi\colon \ker D^\top \to \mathbb{R}^{m-n}$. Now denote $S := \Psi\big(\mathrm{STSP}(n) - x^*\big)$. This is a full-dimensional polytope in $\mathbb{R}^{m-n}$ which contains $\mathbf{0}$ as an interior point. It is affinely isomorphic to STSP(n). Let $S^\vartriangle := \{a \in \mathbb{R}^{m-n} \mid \forall x \in S\colon ax \leq 1\}$ be the polar polytope of S.

[3]We direct the reader to a variant in our terminology: a "combinatorial isomorphism" establishes the fact that two polytopal complexes are what is frequently called "combinatorially equivalent".

standard scaling **5.5.1 Definition** Let (a, α) be an inequality which is valid for STSP(n). If $ax^* - \alpha = 1$, we say that (a, α) is *in standard scaling* with respect to x^* (we will omit mentioning the x^*).

It is obvious that the standard scaling is unique in $\mathbb{R}^*_+ \cdot (a, \alpha)$ and that if a valid inequality (a, α) is scaled by $1/(ax^* - \alpha)$, then the resulting inequality is in standard scaling. The following trivial lemma is the reason for the construction. We let $P^- := -\Psi \circ P$. P^-

5.5.2 Lemma *If (a, α) is in standard scaling then $P^- a \in S^\Delta$. If F is the face of STSP(n) defined by (a, α), then $\Psi(F - x^*)$ is the face of S corresponding to $P^- a$.*

Proof. For every $y := x - x^*$ with $x \in$ STSP(n), the following simple calculation shows that the slack of the inequality $\langle P^- a \mid \Psi y \rangle \leq 1$ is equal to the slack of the inequality $ax \geq \alpha$.

$$\begin{aligned} 1 - \left\langle P^- a \mid \Psi y \right\rangle = 1 + \langle Pa \mid y \rangle &= 1 + \langle Pa + (\mathrm{Id} - P)a \mid y \rangle - \langle (\mathrm{Id} - P)a \mid y \rangle \\ &= 1 + \langle a \mid y \rangle = 1 - ax^* + ax = ax - \alpha. \end{aligned}$$

□

Now let F be a good face of STSP(n), let $\{H^0, \dots, H^k\}$ be the set of NR-facets containing F and let H^j be defined by the TT-form inequality (a_j, α_j) *in standard scaling.* With A as defined in (5.5), let $L \subseteq \mathbb{R}^{k+1} \times \mathbb{R}^{V_n}$ be the vector space of solutions (ϑ, ξ) of the linear system $A\vartheta - \binom{D}{\mathbf{1}^\top}\xi = \mathbf{0}$. With $\mathrm{pr}_1 \colon (\vartheta, \xi) \mapsto \vartheta$, denote by $\Theta := \mathrm{pr}_1(L)$ the projection of L to $\mathbb{R}^{k+1}$. Since the zero-vector $\mathbf{0}$ is in TT-form, for every $(\vartheta, \xi) \in L$ we have $(\vartheta, \lambda(\vartheta)) \in L$. Since $\binom{D}{\mathbf{1}^\top}$ has full row rank, this implies $\xi = \lambda(\vartheta)$ for all $(\vartheta, \xi) \in L$, whence we have $L = \mathrm{Gr}_\Theta\, \lambda$. The same argument yields, by the following simple calculation, that λ is linear on Θ: Θ

$$\begin{aligned} A(\eta_1\vartheta_1 + \eta_2\vartheta_2) - \tbinom{D}{\mathbf{1}^\top}\lambda(\eta_1\vartheta_1 + \eta_1\vartheta_2) = \mathbf{0} &= \eta_1 \left(A\vartheta_1 - \tbinom{D}{\mathbf{1}^\top}\lambda(\vartheta_1)\right) + \eta_2 \left(A\vartheta_2 - \tbinom{D}{\mathbf{1}^\top}\lambda(\vartheta_2)\right) \\ &= A(\eta_1\vartheta_1 + \eta_2\vartheta_2) - \tbinom{D}{\mathbf{1}^\top}(\eta_1\lambda(\vartheta_1) + \eta_2\lambda(\vartheta_2)) \end{aligned}$$

The following lemma makes clear which role the space Θ plays in the construction of non-NR facets.

5.5.3 Lemma *Let $\mu, \nu \in \mathbb{R}^{k+1}$. For an $\eta \in \mathbb{R}$, if $(c_\mu, \gamma_\mu) = \eta(c_\nu, \gamma_\nu)$, then $\mu = \eta\nu$ modulo Θ, i.e., $\mu + \Theta = \eta\nu + \Theta$.*

For $\eta > 0$ the reverse implication holds: if $\mu = \eta\nu$ modulo Θ then $(c_\mu, \gamma_\mu) = \eta(c_\nu, \gamma_\nu)$.

Note that we do not require $\mu, \nu \in \mathbb{R}^{k+1}_+$.

Proof. In the first situation, we have

$$\begin{aligned} \mathbf{0} = (c_\mu, \gamma_\mu) - \eta \cdot (c_\nu, \gamma_\nu) &= A\mu - \tbinom{D}{\mathbf{1}^\top}\lambda(\mu) - \eta A\nu + \eta\tbinom{D}{\mathbf{1}^\top}\lambda(\nu) \\ &= A(\mu - \eta\nu) - \tbinom{D}{\mathbf{1}^\top}(\lambda(\mu) - \eta\lambda(\nu)) = A(\mu - \eta\nu) - \tbinom{D}{\mathbf{1}^\top}\lambda(\mu - \eta\nu). \end{aligned}$$

Note that the last equation holds because the left hand side is in TT-form. If, on the other hand, $\mu = \eta\nu \mod \Theta$ with $\eta > 0$, then letting $\vartheta := \eta\nu - \mu$ we have $A\vartheta - \binom{D}{\mathbf{1}^\top}\lambda(\vartheta) = \mathbf{0}$ and hence

$$\begin{aligned} (c_\mu, \gamma_\mu) &= A\mu - \tbinom{D}{\mathbf{1}^\top}\lambda(\mu) + A\vartheta - \tbinom{D}{\mathbf{1}^\top}\lambda(\vartheta) \\ &= A(\eta\nu) - \tbinom{D}{\mathbf{1}^\top}(\lambda(\mu) + \lambda(\vartheta)) \underset{(*)}{=} A(\eta\nu) - \tbinom{D}{\mathbf{1}^\top}(\lambda(\eta\nu)) \underset{(**)}{=} \eta\left(A\nu - \tbinom{D}{\mathbf{1}^\top}\lambda(\nu)\right) \\ &= \eta\,(c_\nu, \gamma_\nu), \end{aligned}$$

where the equation $(*)$ holds because c_μ is in TT-form (cf. 5.2.11), and equation $(**)$ holds because λ is positive homogeneous by Lemma 5.4.3. □

Let $\pi\colon \mathbb{R}^{k+1} \to \mathbb{R}^{k+1}/\Theta$ denote the canonical projection mapping. We define the following mapping

$$\rho\colon \mathbb{R}^{k+1} \to \operatorname{lin} F^{\Diamond}\colon \mu \mapsto \sum_{j=0}^{k} \mu_j P^- a_j.$$ ρ

5.5.4 Lemma *Let $\mathbb{A}^k/\Theta := \pi(\mathbb{A}^k)$ denote the image of $\mathbb{A}^k$ under π. There exists an affine isomorphism $\Phi\colon \mathbb{A}^k/\Theta \to \operatorname{aff} F^{\Diamond}$ which maps the projection $\mathbb{A}^k/\Theta := \pi(\mathbb{A}^k)$ of the k-dimensional standard simplex $\mathbb{A}^k$ onto $F^{\Diamond}$, i.e., the two polytopes are affinely isomorphic. Furthermore, the following diagram commutes:* Φ

$$\begin{tikzcd}
\mathbb{A}^k/\Theta \arrow[r, "\Phi"] & \operatorname{aff}(F^{\Diamond}) \\
\mathbb{A}^k \arrow[u, "\pi"] \arrow[ur, "\rho"'] &
\end{tikzcd}$$

This means that we have $\Phi \circ \pi = \rho$ on $\mathbb{A}^k$, or, in other words, $\Phi(\pi(\mu)) = \sum_j \mu_j P^- a_j$ for all $\mu \in \mathbb{A}^k$.

Proof. Consider the mapping

$$\rho_1\colon \mathbb{R}^{k+1} \to \operatorname{lin}(F^{\Diamond} \times 1)\colon \mu \mapsto \sum_{j=0}^{k} \mu_j \begin{pmatrix} P^- a_j \\ 1 \end{pmatrix}.$$

Claim 1. $\ker \rho_1 \subseteq \Theta$.

If the claim is is true, there exists a mapping $\Phi_1\colon \mathbb{R}^{k+1}/\Theta \to \operatorname{lin}(F^{\Diamond} \times 1)$ with the property that $\Phi_1(\pi(\mu)) = \sum_j \mu_j \left(\begin{smallmatrix} P^- a_j \\ 1 \end{smallmatrix}\right)$, i.e., the following diagram commutes:

$$\begin{tikzcd}
\mathbb{R}^{k+1}/\Theta \arrow[r, "\Phi_1"] & \operatorname{lin}(F^{\Diamond} \times 1) \\
\mathbb{R}^{k+1} \arrow[u, "\pi"] \arrow[ur, "\rho_1"'] &
\end{tikzcd}$$

The next step in the proof is then the following:

Claim 2. Φ_1 is a linear isomorphism. Or, equivalently, $\ker \rho_1 = \Theta$.

If the second claim is true, we can complete the proof of the lemma by noting that Φ_1 maps $\mathbb{A}^k/\Theta$ onto $\operatorname{aff}(F^{\Diamond} \times 1)$, which is easily seen from the equation

$$\Phi_1(\mathbb{A}^k/\Theta) = \rho_1(\pi^{-1}(\mathbb{A}^k/\Theta)) = \rho_1(\mathbb{A}^k + \Theta) \underset{(*)}{=} \rho_1(\mathbb{A}^k) = \operatorname{aff}(F^{\Diamond} \times 1).$$

Note that we used $\Theta \subseteq \ker \rho_1$ for $(*)$. The last detail in the proof, namely that $\operatorname{aff}(F^{\Diamond} \times 1) \cong \operatorname{aff} F^{\Diamond}$, hardly deserves mentioning. We define Φ to be the composition of Φ_1 with this isomorphism.

$$\begin{tikzcd}
\mathbb{A}^k/\Theta \arrow[r, shift left, "\Phi_1"] \arrow[rr, bend left, "\Phi"] & \operatorname{aff}(F^{\Diamond} \times 1) \arrow[l, shift left] \arrow[r, leftrightarrow, "\cong"] & \operatorname{aff} F^{\Diamond} \\
& \mathbb{A}^k \arrow[ul, "\pi"] \arrow[u, "\rho_1"'] \arrow[ur, "\rho"'] &
\end{tikzcd}$$

Now we prove the claims.

Proof of Claim 1. Let $\vartheta \in \ker \rho_1$, i.e., $\sum_j \vartheta_j P^- a_j = \mathbf{0}$ and $\sum_j \vartheta_j = 0$. The first equation is equivalent to $\sum_j \vartheta_j a_j \in \ker P^-$ but clearly $\ker P^- = \ker P = (\operatorname{Im} P)^\perp = \operatorname{Im} D$. Thus there exists $\xi \in \mathbb{R}^{V_n}$ with

$$\sum_{j=0}^{k} \vartheta_j a_j - D\xi = \mathbf{0}.$$

This means that there exists an $\alpha \in \mathbb{R}$ with $A\vartheta - \left(\begin{smallmatrix} D \\ \mathbf{1}^\top \end{smallmatrix}\right)\xi = \left(\begin{smallmatrix} \mathbf{0} \\ \alpha \end{smallmatrix}\right)$. To prove $\vartheta \in \Theta$ it suffices to show that $\alpha = 0$. This follows of course from the second equation $\sum_j \vartheta_j = 0$ above: since the (a_j, α_j) are in standard scaling with respect to x^*, we have $0 = \sum_j \vartheta_j \cdot 1 = (\sum_j \vartheta_j a_j)x^* - \sum_j \vartheta_j \alpha_j$. This means that $A\vartheta$ and hence also $A\vartheta - \left(\begin{smallmatrix} D \\ \mathbf{1}^\top \end{smallmatrix}\right)\xi$ is satisfied by x^* with equality, which implies $\alpha = 0$.

Proof of Claim 2. This proof can be done in two ways, the first being simply to understand that $\Theta \subseteq \ker \rho_1$ holds because applying $\left(\begin{smallmatrix} P^- & 0 \\ x^{*\top} & -1 \end{smallmatrix}\right)$ to the equation $A\vartheta - \left(\begin{smallmatrix} D \\ \mathbf{1}^\top \end{smallmatrix}\right)\xi = \mathbf{0}$ yields $\sum_j \vartheta_j \left(\begin{smallmatrix} P^- a_j \\ 1 \end{smallmatrix}\right) = \mathbf{0}$. The second way to prove Claim 2 is by comparing the dimensions. Since the rank of $\left(A \;\; \begin{smallmatrix} D \\ \mathbf{1}^\top \end{smallmatrix}\right)$ is equal to $m - \dim F = n + \operatorname{codim} F$, we have

$$\dim \Theta = \dim \ker \left(A \quad \begin{smallmatrix} D \\ \mathbf{1}^\top \end{smallmatrix}\right) = k + 1 + n - (n + \operatorname{codim} F) = k + 1 - \operatorname{codim} F.$$

On the other hand, since $\dim F^\Diamond = \operatorname{codim} F - 1$, we have $\dim \operatorname{lin}(F^\Diamond \times 1) = \operatorname{codim} F = \dim \mathbb{R}^{k+1}/\Theta$, whence it follows that Φ_1 is an isomorphism. $\square$

5.5.5 Corollary *Θ is parallel to $\mathbb{A}^k$, i.e., $\mathbf{1}\vartheta = 0$ holds for all $\vartheta \in \Theta$.*

Proof. One way to see this is to use the fact that $\Theta = \ker \rho_1$, as defined and proved in the proof of the previous lemma: clearly $\vartheta \in \ker \rho_1$ implies $\sum_j \vartheta_j = 0$. Another way to prove the corollary is to use the statement of the lemma, because, with $\mathbb{L}^k := \mathbb{A}^k - 1$, we have

$$\begin{aligned} k - \dim(\Theta \cap \mathbb{L}^k) = \dim \mathbb{A}^k/\Theta = \dim F^\Diamond &= \operatorname{codim} F - 1 \\ &= k + 1 - \dim \ker \left(A \;\; \begin{smallmatrix} D \\ \mathbf{1}^\top \end{smallmatrix}\right) - 1 = k - \dim \Theta. \end{aligned}$$

$\square$

5.5.6 Corollary *Let $\mu, \nu \in \mathbb{A}^k$. If (c_μ, γ_μ) and (c_ν, γ_ν) are collinear, then $\mu = \nu \mod \Theta$.*

Proof. Suppose that $(c_\mu, \gamma_\mu) = \eta(c_\nu, \gamma_\nu)$. From Lemma 5.5.3, we know that $\mu - \eta\nu \in \Theta$. Now, by Corollary 5.5.5, we conclude that

$$0 = \langle \mathbf{1} \mid \mu - \eta\nu \rangle = \mathbf{1}\mu - \eta\mathbf{1}\nu = 1 - \eta,$$

whence $\eta = 1$, i.e., $\mu - \nu \in \Theta$. $\square$

5.5.7 Proposition *The mapping $\mu \mapsto (c_\mu, \gamma_\mu)\colon \mathbb{A}^k \to \mathbb{R}^m \times \mathbb{R}$ factors along ρ, i.e., there exists a mapping*

$(\bar c, \bar\gamma)$

$$(\bar c, \bar\gamma)\colon \quad \operatorname{aff} F^\Diamond \to \mathbb{R}^m \times \mathbb{R}\colon \quad a \mapsto \big(\bar c(a), \bar\gamma(a)\big)$$

with the property that $(\bar c, \bar\gamma) \circ \rho = (c_\centerdot, \gamma_\centerdot)$, i.e., the following diagram commutes

$$\begin{tikzcd} \operatorname{aff} F^\Diamond \arrow[r, "(\bar c,\bar\gamma)"] & \mathbb{R}^m \times \mathbb{R} \\ \mathbb{A}^k \arrow[u, "\rho"] \arrow[ur, "(c_\centerdot,\gamma_\centerdot)"'] & \end{tikzcd}$$

In other words, for all $\mu \in \mathbb{A}^k$ we have

$$\Big(\bar c\big(\textstyle\sum_j \mu_j P^- a_j\big) \,,\; \bar\gamma\big(\textstyle\sum_j \mu_j P^- a_j\big) \Big) = (c_\mu, \gamma_\mu).$$

Furthermore, if $a, b \in \operatorname{aff} F^\Diamond$ such that $\big(\bar c(a), \bar\gamma(a)\big)$ and $\big(\bar c(b), \bar\gamma(b)\big)$ are collinear, then $a = b$.

Proof. From Lemma 5.5.3 for $\mu, \nu \in \mathbb{A}^k$ we know that if $(c_\mu, \gamma_\mu) = (c_\nu, \gamma_\nu)$, then $\mu - \nu \in \Theta$, i.e., $\pi(\mu) = \pi(\nu)$. Since $\rho = \Phi \circ \pi$ on $\mathbb{A}^k$, this implies $\rho(\mu) = \rho(\nu)$. From this, the existence of $(\bar{c}, \bar{\gamma})$ follows.

The last statement follows from Corollary 5.5.6. □

5.5.2 A subdivision of $F^\diamondsuit$

5.5.8 Lemma *Let $u \in V_n$. If $\mu, \nu \in \mathbb{A}^k$ with $\rho(\mu) = \rho(\nu)$ and $e \not\ni u$ then $t_{u,e}(\mu) = \lambda_u(\mu)$ if and only if $t_{u,e}(\nu) = \lambda_u(\nu)$.*

Proof. This follows directly from Proposition 5.5.7, since $t_{u,e}(\mu) = \lambda_u(\mu)$ is equivalent to $e \in \Delta_u(c_\mu) = \Delta_u(\bar{c}(\rho(\mu)))$. □

This lemma justifies the following key definition (recall the definition of $\mathscr{F}(\cdot)$ from page 2).

5.5.9 Definition Let $u \in V_n$. For $a = \rho(\mu) \in F^\diamondsuit$ we define a set of edges

$E_u(a)$

$$E_u(\rho(\mu)) := \{e \not\ni u \mid t_{u,e}(\mu) = \lambda_u(\mu)\},$$

and the following subsets of $F^\diamondsuit$

$P_u(a)$
$P_u^\circ(a)$

$$P_u(a) := \{b \in F^\diamondsuit \mid \mathscr{F}(b) \supseteq \mathscr{F}(a) \;\wedge\; E_u(b) \supseteq E_u(a)\}$$
$$P_u^\circ(a) := \{b \in F^\diamondsuit \mid \mathscr{F}(b) = \mathscr{F}(a) \;\wedge\; E_u(b) = E_u(a)\}.$$

5.5.10 Lemma *Let $u \in V_n$ and $a \in F^\diamondsuit$.*

(a). $P_u(a)$ is a polytope.

(b). $P_u^\circ(a) = \operatorname{relint} P_u(a)$, i.e., $P_u^\circ(a)$ is the relative interior of $P_u(a)$.

(c). Let $a, b \in F^\diamondsuit$. The following statements are equivalent:

(i) $P_u(a) = P_u(b)$

(ii) a is a relative interior point of $P_u(b)$

(iii) b is a relative interior point of $P_u(a)$.

(d). If P' is a face of $P_u(a)$, and b a relative interior point of P', then $P' = P_u(b)$.

(e). For $a, b \in F^\diamondsuit$, the intersection $P_u(a) \cap P_u(b)$ is a face of both $P_u(a)$ and $P_u(b)$.

Proof. *(a)* Let $a = \rho(\mu)$, and $P := P(a) := \rho(Q(a))$ where

$$Q := Q(a) := \left\{\nu \in \mathbb{A}^k \;\middle|\; \forall e \in E_u(a)\; \forall f \not\ni u \colon t_{u,e}(\nu) \le t_{u,f}(\nu)\right\}. \tag{5.6}$$

Q is the intersection of $\mathbb{A}^k$ with the set of solutions to a system of linear inequalities. Thus, Q and P are polyhedra. We show that $P_u(a) = P \cap F^\diamondsuit \cap \bigcap_{X \in \mathscr{F}(a)} X$. Let $\rho(\nu) = b \in P \cap F^\diamondsuit \cap \bigcap_{X \in \mathscr{F}(a)} X$. Clearly $\mathscr{F}(b) \supseteq \mathscr{F}(a)$, and for all $e \in E_u(a)$ we have $t_{u,e}(\nu) = \lambda_u(\nu)$, i.e., by Lemma 5.5.8, $e \in E_u(b)$. On the other hand, let $\rho(\nu) = b \in P_u(a)$. Then, obviously, $b \in F^\diamondsuit \cap \bigcap_{X \in \mathscr{F}(a)} X$. But for $e \in E_u(a)$ we also have $t_{u,e}(\nu) = \lambda_u(\nu)$ and hence $t_{u,e}(\nu) \le t_{u,f}(\nu)$ for all $f \not\ni u$, i.e., $\nu \in Q$ and hence $b \in P$.

(b) We will use the notation of (a). Additionally, for all $a \in F^\diamondsuit$, $e \in E_u(a)$ and $f \not\ni u$, define

$$Q_{e,f} := Q_{e,f}(a) := \{\nu \in Q(a) \mid t_{u,e}(\nu) = t_{u,f}(\nu)\}.$$

Now note that either $Q_{e,f}$ is empty, or Θ is contained in the linear space defined by $Q_{e,f}$, since, if $\nu_0 \in Q_{e,f}$, then $t_{u,f}(\nu_0) = \lambda_u(\nu_0) = t_{u,e}(\nu_0)$, and, by Lemma 5.5.8, $t_{u,f}(\nu) = \lambda_u(\nu) = t_{u,e}(\nu)$, for all $\nu \in \nu_0 + \Theta$. Hence, from $\rho = \Phi \circ \pi$, we see that every facet of P is the image under ρ of a facet of Q. Therefore, $b = \rho(\nu) \in F^\diamondsuit$ is in the relative interior of $P_u(a)$ iff the following three conditions hold: $\mathscr{F}(b) = \mathscr{F}(a)$, $t_{u,e}(\nu) = t_{u,f}(\nu)$ for all $e, f \in E_u(a)$ and for every $f \not\ni u$, $f \notin E_u(a)$ we have $t_{u,e}(\nu) < t_{u,f}(\nu)$. This is equivalent to $b \in P_u^\circ(a)$.

(c) It is clear by (b) that both (ii) and (iii) imply (i). On the other hand, $P_u(a) = P_u(b)$ implies $E_u(a) = E_u(b)$ and $\mathscr{F}(b) = \mathscr{F}(a)$, whence (ii) and (iii) follow.

(d) The proof can be carried on by induction on $\operatorname{codim} P'$. We thus assume that P' is a facet of $P_u(a)$. Since every facet of P is the image under ρ of a facet of Q, P' is the intersection with $P_u(a)$ of facets of $F^\diamondsuit$ and of images of sets $Q_{e,f}$ with $e \in E_u(a)$. The set of facets of $F^\diamondsuit$ which contain P' is of course just $\mathscr{F}(b)$, and $P' \subseteq \rho(Q_{e,f})$ holds iff $t_{u,f}(\mu) = t_{u,e}(\mu) = \lambda_u(\mu)$, where $\mu \in \mathbb{\Delta}^k$ is chosen such that $\rho(\mu) = b$. Consequently for $c \in F^\diamondsuit$ we have $c \in P'$ iff $\mathscr{F}(c) \supseteq \mathscr{F}(b)$ and $E_u(c) \supseteq E_u(b)$, which proves the assertion.

(e) Suppose that $P_u(a) \neq P_u(b)$. Then, by (b), $P_u(a) \cap P_u(b)$ cannot contain a relative interior point of both $P_u(a)$ and $P_u(b)$. □

We summarize the main statement of the lemma in the following proposition.

$\mathcal{T}_u$

5.5.11 Proposition *The set* $\mathcal{T}_u(F) := \{\emptyset\} \cup \{P_u(a) \mid a \in F^\diamondsuit\}$ *is a pure (finite) polytopal complex of dimension* $\dim F^\diamondsuit$ *whose underlying space is* $F^\diamondsuit$.

Proof. The finiteness follows from the fact that there are only finitely many possible choices for the sets $\mathscr{F}(\cdot)$ and $E_u(\cdot)$.

The fact that the complex is pure follows from elementary topological arguments (like that the complement in $F^\diamondsuit$ of the union of finitely many polytopes of dimension $\dim F^\diamondsuit$ is an open subset of $F^\diamondsuit$, hence empty or of dimension $F^\diamondsuit$, and that the union of finitely many polytopes of dimension strictly less than $\dim F^\diamondsuit$ is nowhere-dense in an open subset of $F^\diamondsuit$). □

tilting complex $\mathcal{T}$

5.5.12 Definition The polytopal complex $\mathcal{T}(F)$ which is defined as the set of all intersections of members of $\mathcal{T}_u(F)$, $u \in V_n$, the union of which is $F^\diamondsuit$. It is a subdivision of $F^\diamondsuit$. We call $\mathcal{T}(F)$ the *tilting complex* for F.

5.5.13 Lemma *Let* $P \in \mathcal{T}(F)$ *and* $a \in \operatorname{relint} P$. *Then* $P = \bigcap_u P_u(a)$.

Proof. Let $P_u(b) \supseteq P$. Then either P is contained in a face of $P_u(b)$, in which case we do not need to look at $P_u(b)$, or $\operatorname{relint} P \subseteq \operatorname{relint} P_u(b)$, whence $P_u(b) = P_u(a)$ by Lemma 5.5.10.(b). □

5.5.3 Looking at $B(n)$

TT-type face of $B(n)$

It will be convenient to expand the term "TT-type" to faces of the blocking polyhedron $B(n)$. We say that a non-empty face X of $B(n)$ is of TT-type if it is not trivial in the sense of Proposition 0.1.1, and $X^\sharp$ (cf. Proposition 0.1.1) is a TT-type face of GTSP(n). Equivalently, a non-empty face X is of TT-type, if it is bounded and does not contain a degree vertex. Consequently, we call a vertex a of $B(n)$ an *NR-vertex,* if $\{a\}$ is a TT-type face, and the face $x \vee \bigvee_{u \in V_n} \delta_u$ of $B(n)$ is a simplex.

5.5.14 Lemma *Let* $P \in \mathcal{T}(F)$ *be a polytope in the tilting complex.* $(\bar{c}, \bar{\gamma})$ *is affine on* P.

Proof. We first show that λ_u is linear on $\rho^{-1}(P_u(a)) \cap \mathbb{\Delta}^k$ for all u, a. To see this, let $e \in E_u(a)$, and note that for all $\mu \in \mathbb{\Delta}^k$ with $\rho(\mu) \in P_u(a)$ we have $t_{u,e}(\mu) = \lambda_u(\mu)$.

The linearity of λ_u for all u on $\rho^{-1}(P_u(a)) \cap \mathbb{\Delta}^k$ implies that λ is linear on $\rho^{-1}(P) \cap \mathbb{\Delta}^k$, for $P \in \mathcal{T}(F)$. Hence, $(c_\centerdot, \gamma_\centerdot)$ is linear on $\rho^{-1}(P) \cap \mathbb{\Delta}^k$. Since $(c, \gamma) \circ \rho = (c_\centerdot, \gamma_\centerdot)$, the statement of the lemma follows. □

5.5.15 Lemma $\bar{\gamma}$ *never vanishes, or, in other words,* $(\bar{c}, \bar{\gamma})\,(F^\diamondsuit) \subseteq \mathbb{R}^{E_n} \times \mathbb{R}_+^*$.

Proof. Suppose $\bar{\gamma}(\rho(\mu)) = 0$. Then either $c_\mu = 0$, whence $F = \mathrm{STSP}(n)$, or (c_μ, γ_μ) is the sum of non-negativity inequalities. Both conclusions contradict the fact that F is a good face. □

We now consider the continuous mapping $\varphi := \bar{c}/\bar{\gamma}$, i.e.,

$$\varphi\colon F^{\diamondsuit} \to B(n)\colon a \mapsto \frac{1}{\bar{\gamma}(a)}\,\bar{c}(a).$$

The mapping φ is piecewise projective, or to be precise, it is projective on every polytope in $\mathcal{T}$. A consequence of this is the following fact, for which we give a direct proof.

5.5.16 Lemma *If $P \in \mathcal{T}(F)$ then $\varphi(P)$ is a polytope of dimension* $\dim P$ *in* $\mathbb{R}^{E_n}$.

Proof. By Lemma 5.5.14 $P' := (\bar{c}, \bar{\gamma})\,(P)$ is a polytope, which, since $(\bar{c}, \bar{\gamma})$ is injective, has dimension $\dim P$. By the second part of Proposition 5.5.7, the cone $C := \mathbb{R}_+ \cdot P'$ has dimension $\dim P + 1$ and is contained in $\mathbb{R}^{E_n} \times \mathbb{R}_+^* \cup \{\mathbf{0}\}$ by Lemma 5.5.15. Consequently, $C \cap \mathbb{R}^{E_n} \times \{1\}$ and hence $\varphi(P)$ have dimension $\dim P$. □

We note the following direct consequence of the second part of Proposition 5.5.7 explicitly.

5.5.17 Lemma *φ is injective.* □

5.5.18 Remark For all $P \in \mathcal{T}(F)$ we have $\varphi(\operatorname{relint} P) = \operatorname{relint} \varphi(P)$. This follows from topological or geometric ingredients, as you prefer.

For the following lemma, for an $a \in F^{\diamondsuit}$, we define $\operatorname{ham}(a)$ to be the set of all $x \in F \cap \mathscr{H}(n)$ which, in the order of the face-lattice of the polar of STSP(n), are greater than or equal to all $X \in \mathscr{F}(a)$. If $a \in \operatorname{relint} P$, we let $\operatorname{ham}(P) := \operatorname{ham}(a)$. $\operatorname{ham}(a)$, $\operatorname{ham}(P)$

5.5.19 Lemma *Let $P \in \mathcal{T}(F)$. For all $a \in$* $\operatorname{relint} P$ *the same face of GTSP(n) is defined by* $(\varphi(a), 1)$.

Proof. By Lemma 5.5.13 we have $P = \bigcap_u P_u(a)$. Hence

$$\operatorname{relint} P = \bigcap_u \operatorname{relint} P_u(a) = \bigl\{b \in F^{\diamondsuit} \mid \mathscr{F}(b) = \mathscr{F}(a)\bigr\} \;\cap\; \bigcap_u \bigl\{b \in F^{\diamondsuit} \mid E_u(b) = E_u(a)\bigr\}.$$

Now $\mathscr{F}(a)$ determines the set of vertices of STSP(n) which are satisfied with equality by the inequality $(\varphi(a), 1)$, namely this is just the set $\operatorname{ham}(a)$. The same holds for $\mathscr{F}(b)$ and $(\varphi(b), 1)$. Further, $E_u(a) = \Delta_u(\varphi(a))$, and the same holds for b. Hence the assertion of the lemma follows by Lemma 5.3.6. □

The main result of this section is the following theorem.

5.5.20 Theorem *$\varphi\colon F^{\diamondsuit} \to B(n)$ induces a combinatorial isomorphism of the polytopal complex $\mathcal{T}(F)$ to the subcomplex of the boundary complex of $B(n)$ which consists of all TT-type faces of $B(n)$ which are contained in $F^{\sharp}$ (or, in other words, it is the image under $\square^{\sharp}$ of all all TT-type faces of GTSP(n) containing F). For all $P = \bigcap_{u \in U} P_u(a) \in \mathcal{T}(F)$ (cf. Lemma 5.5.13), $\varphi(P)^{\sharp}$ is the unique face G of GTSP(n) which satisfies $G \cap \mathscr{H}(n) = \operatorname{ham}(a)$, and $\Delta_u(G) = E_u(a)$ for all u.*

Proof. The mapping $P \mapsto \varphi(P)$ clearly defines a combinatorial isomorphism of some polytopal complexes. We have to make sure that $\varphi(P)$ is a member of the boundary complex of $B(n)$ for all P, i.e., we show that for $P \in \mathcal{T}(F)$, the image $\varphi(P)$ is a face of $B(n)$.

We claim that for all P the image $\varphi(P)$ is contained in a face of $B(n)$ with dimension $\dim P$. The proof of the claim is by induction on $\dim F^{\diamondsuit} - \dim P$. Note that we rely on the fact that the tilting complex $\mathcal{T}(F)$ is pure.

If $\dim P = \dim F^{\diamondsuit}$, it follows by Lemma 5.4.5 that $\varphi(P)$ is contained in a union of $\dim P$-dimensional faces of $B(n)$. By Lemma 5.5.19, we know that $\varphi(P)$ is entirely contained in all faces which intersect $\operatorname{relint} \varphi(P) = \varphi(\operatorname{relint} P)$, and by Lemma 5.5.16, we know that a face containing $\varphi(P)$ has dimension at least $\dim P$. Hence $\varphi(P)$ is contained in a face of dimension $\dim P$ of $B(n)$.

Now suppose that P' is contained in a face Q' of $B(n)$ with $\dim P' = \dim Q'$, and let P be a facet of P'. Let $a' \in \operatorname{relint} P'$ and $a \in \operatorname{relint} P$ and let $R_{a'}, R_a$ denote the face of GTSP(n) defined by $(\varphi(a'), 1)$, $(\varphi(a), 1)$. Then $\mathscr{F}(a) \supseteq \mathscr{F}(a')$ and $\forall u\colon E_u(a) \supseteq E_u(a')$ holds, but, by Lemma 5.5.10-(b), at least one of $\mathscr{F}(a) \supsetneq \mathscr{F}(a')$ or $\exists u\colon E_u(a) \supsetneq E_u(a')$ is true. This means that there exists an $x \in R_a \setminus R_{a'}$, whence $R_{a'}$ is a proper face of R_a. Consequently $\varphi(a)$ is a relative interior point of a face Q which is a proper face of Q'. By Lemma 5.5.19, $\varphi(P)$ is contained in Q and by Lemma 5.5.16, Q must be a facet of Q'.

This completes the proof of the claim.

The claim implies that for $P \in \mathcal{T}(F)$, the image $\varphi(P)$ is actually equal to a face of $B(n)$ by the topological invariance of the boundary (or use a geometric argument if you must). $\square$

5.5.21 Corollary *The image under $\square^\sharp$ of the set all TT-type faces of GTSP(n) which contain F form a pure subcomplex of dimension* $\operatorname{codim} F - 1$ *of the boundary complex of $B(n)$. Topologically, it is a closed ball.* $\square$

5.6 An algorithmic perspective

We now sketch how $\mathcal{T}(F)$ can be constructed algorithmically. For this we use a different approach to the construction of $\mathcal{T}(F)$ which is based on the following proposition.

5.6.1 Proposition *$\mathcal{T}_u(F)$ is the image under ρ of the regular subdivision of the simplex $\mathbb{A}^k$ which can be obtained from the piecewise linear concave tilting function λ_u.*

Proof. Picking up the notation of the proof of Lemma 5.5.10 on page 67, it is apparent from the definitions that λ_u is linear on each polyhedron $Q := Q(a)$. But as $P_u(a) = F^\diamondsuit \cap P \cap \bigcap_{X \in \mathscr{F}(a)} X$, and $F^\diamondsuit \cap P = \rho(\mathbb{A}^k \cap Q)$, we see that the subdivision of $\mathbb{A}^k$ defined by λ_u is just $\mathcal{T}_u(F)$. $\square$

Algorithm 5.2 Tilting complex

```
Input:
    All STSP-facet defining inequalities $(a_j, \alpha_j)$, $j = 0, \dots, k$, containing a good face $F$ in
    TT-form.
Output:
    Polytopal complex $\mathcal{T}$ with union $\mathbb{A}^k$ and linear space $\Theta$ with $\mathcal{T}/\Theta \cong \mathcal{T}(F)$.
1: Scale $(a_j, \alpha_j)$ into standard scaling with respect to $x^* := 1/(n-1)\,\mathbf{1}$.
2: Compute the rank $r := \operatorname{rk}\begin{pmatrix} A & D \\ & \mathbf{1}^\top \end{pmatrix}$ and $\Theta$.
3: Compute the triangle defects $\bar{t}^j := \bar{t}(a_j)$, define $t^*_{u,e} := (\bar{t}^0_{u,e}, \dots, \bar{t}^k_{u,e})^\top$.
4: For all $u$, let $T'_u$ be a maximal set of edges $e \not\ni u$ such that the $t^*_{u,e}$, $e \in T'_u$, are distinct.
5: Build the sets $T_u$ consisting of all $e \in T'_u$ for which there exists no $\zeta \geq 0$ with
       $$\sum_{f \neq e} \zeta_f \begin{pmatrix} -t^*_{u,f} \\ 1 \end{pmatrix} \leq \begin{pmatrix} -t^*_{u,e} \\ 1 \end{pmatrix}.$$
6: Compute the face lattices of the polytopes
   $Q_u := \{(\mu, \lambda) \in \mathbb{R}^{k+1} \times \mathbb{R} \mid \mu \in \mathbb{A}^k \wedge 0 \leq \lambda \leq \mu^\top t^*_{u,e}\}$.
7: Using trivial Fourier-Motzkin elimination, project all proper faces of the $Q_u$ onto $\mu$-space,
   obtaining a system similar to (5.6).
8: Again using Fourier-Motzkin elimination, project all faces onto the orthogonal complement
   $\Theta^\perp$ of the linear space $\Theta$.
9: For each $n$-tuple of projected faces, compute their intersection by solving an LP-feasibility
   problem and Gaussian elimination.
```

Algorithm 5.2 gives a rough sketch of the construction of $\mathcal{T}(F)$ using this proposition. It assumes the (a_j, α_j), $j = 0, \dots, k$, as input. The output is a subdivision of $\mathbb{A}^k/\Theta$, which is affinely isomorphic to the tilting complex $\mathcal{T}(F)$. Here, the algorithm relies on the fact that $\mathbb{A}^k/\Theta \cong F^\diamond$ from Lemma 5.5.4. There may be more elaborate algorithms to compute $\mathcal{T}(F)$. With this algorithm we intend to illuminate the theory of tilting complexes from a different perspective.

5.7 Application to computation of complete descriptions of GTSP polyhedra

We now give an application of tilting complexes. A computational technique to find a complete list of facets of a polyhedron is the following. Start with a list containing a single (known) facet. While there are "untreated" facets in the list, take one of those, mark it as "treated", compute all facets adjacent to it, and append the adjacent facets to the list if they are not already contained in it. This method is called *adjacency decomposition* in [Chr97, CR01], and it is equivalent to methods enumerating the vertices of a polyhedron by walking along edges.

With today's hardware and software (e.g. Intel Pentium IV, 2.8GHz, PortaForte [Chr98]), and using knowledge of the GTSP-polyhedron, it is possible to compute the neighbors of all TT-type facets of GTSP(9) except for the connectivity facets. It is also possible to compute the neighbors of connectivity facets on STSP(9) [CR96, Chr97]. The question is whether the computation of the neighbors of connectivity facets in GTSP(9) is dispensable. In this section, relying on the theory of tilting complexes, we will answer this question in the affirmative. More generally we establish a relationship between sets of facets which, when interpreted as an LP-relaxation of GTSP(n), satisfy the so-called parsimonious property of Goemans & Bertsimas [GB93] and the structure of the ridge graph of GTSP(n). Recall that the *ridge graph* of a polyhedron P has as its node set the set of all facets of P, and as its edge set the set of ridges of P.

5.7.1 Parsimonious property linked to complete descriptions

An LP-relaxation for GTSP(n) containing all degree inequalities has the *parsimonious property* if, for all metric cost functions, forcing the degree inequalities to equations does not increase the minimum. *parsimonious property*

Let $\mathcal{B}$ be a set of inequalities defining NR-facets of GTSP(n). Suppose that the relaxation of GTSP(n) consisting of non-negativity inequalities, degree inequalities, and $\mathcal{B}$ has the parsimonious property. Let $\mathcal{G}$ denote the graph which results from the ridge graph of GTSP(n) if all nodes corresponding to non-negativity and degree facets and facets defined by inequalities in $\mathcal{B}$ are deleted. $\mathcal{B}$ $\mathcal{G}$

5.7.1 Theorem *Every connected component of $\mathcal{G}$ contains an NR-facet.*

The proof of Theorem 5.7.1 is geometric and invokes the tilting complexes. We first need the following fact.

5.7.2 Lemma *Let $\mathcal{B}$ be a set of inequalities defining NR-facets of GTSP(n). Suppose that the relaxation of GTSP(n) consisting of*

- *all non-negativity inequalities,*
- *all degree inequalities, and*
- *the inequalities in $\mathcal{B}$*

has the parsimonious property. A non-NR facet (c, γ) cannot be written in the form (5.4)*, defined on page 60, with all the $(a_j, \alpha_j) \in \mathcal{B}$.*

Proof. Suppose that (c, γ) can be written as a sum of degree equations and inequalities defining facets in $\mathcal{B}$. Then minimizing the cost function c over the relaxation consisting of degree equations, non-negativity inequalities, and $\mathcal{B}$ produces γ as the minimum. If the degree equations are relaxed to degree inequalities, then, by the parsimonious property, the minimum is still γ. This implies that (c,γ) is dominated by non-negativity and degree inequalities, and inequalities in $\mathcal{B}$. This is impossible since (c,γ) defines a non-NR facet of GTSP(n) and all facets in $\mathcal{B}$ have the NR property. $\square$

We are now ready to prove the main theorem. I am indebted to M. Oswald for a contribution to the proof.

Proof of Theorem 5.7.1. Suppose that F is a face of STSP(n) which is the intersection of a non-NR facet with STSP(n). Let B denote the set of vertices of $F^{\diamondsuit}$ which correspond to facets defined by inequalities in $\mathcal{B}$. We prove that, for every vertex in the graph (1-skeleton) of $\mathcal{T}(F)$, there exists a path connecting it to a vertex of $F^{\diamondsuit}$, which does not use a vertex from B. If $B = \emptyset$, we are done. Otherwise $Q := \operatorname{conv} B \subseteq F^{\diamondsuit}$ is a non-empty polytope. We first note that Lemma 5.7.2 implies that a vertex in $\mathcal{T}(F)$ which is not a member of B cannot be contained in Q.

Let c be a vertex in $\mathcal{T}(F)$ which is not a member of B. If c is a vertex of $F^{\diamondsuit}$, then we are done. Otherwise, we assume w.l.o.g. that c is in the relative interior of $F^{\diamondsuit}$. If that is not the case, instead of F we can treat the face of STSP(n) which is the intersection with STSP(n) with the facet of GTSP(n) defined by $(\varphi(c),1)$. Let (p,π) define a hyperplane separating c from Q, i.e., $\langle q \mid p\rangle < \pi$ for all $q \in Q$, and $\langle c \mid p\rangle > \pi$. Since the union of the faces with codimension at least one in $\mathcal{T}(F)$ is nowhere dense in $F^{\diamondsuit}$, we can assume that p is not parallel to any such face. Hence, there exists an $\varepsilon > 0$ such that the line segment $c +]0,\varepsilon[\cdot p$ is contained in the relative interior of a $\dim F^{\diamondsuit}$-dimensional polytope $P \in \mathcal{T}(F)$, of which c is a vertex. By standard polytope theory, P must have a vertex c' adjacent to c with $\langle c \mid p\rangle < \langle c' \mid p\rangle$. Clearly $c' \notin B$.

To use this argument inductively, we distinguish three cases.

1. If c' is a vertex of $F^{\diamondsuit}$, we are done.
2. Otherwise, if c' is a relative interior point of $F^{\diamondsuit}$ we replace c by c' and use induction on the length of the path.
3. Otherwise, we revert to a face of $F^{\diamondsuit}$ and use induction on $\dim F^{\diamondsuit}$.

This concludes the proof of the theorem. $\square$

5.7.2 Complete description of GTSP(9)

The so-called *subtour relaxation,* i.e., the relaxation consisting of degree inequalities, non-negativity inequalities, and connectivity inequalities, has the parsimonious property for all n [GB93]. In Section 5.4, we prove that there exists a non-NR facet for $n = 9$. In [ORT05], the inequality defining the facet was constructed using a method which we do not repeat here. We obtain the following corollary.

	1	2	3	4	5	6	7	8
0	9	6	6	7	6	10	2	4
1		3	9	6	11	7	7	5
2			6	9	12	10	8	8
3				3	12	10	8	8
4					9	13	9	11
5						4	8	10
6							8	6
7								2

Table 5.3: Coefficients of the unique non-NR facet of GTSP(9), the right hand side is 44.

5.7.3 Corollary *There exist precisely* 9! *non-NR facets of GTSP*(n)*. They can all be obtained by permutation of nodes from the one defined by the inequality shown in Table 5.3.* $\square$

In [ORT05], we also proved that, for $n \leq 8$, there do not exist any non-NR facets of GTSP(n). In order to prove complete descriptions for GTSP(n) for increasing values of n, it will be necessary to check the parsimonious property for other relaxations. The relaxations obtained by the addition of comb inequalities with few teeth is of particular practical importance.

5.8 Intermediate polyhedra

It is intuitively obvious that $\mathcal{T}_u(F)$ is isomorphic as a poset to the poset of all "TT-like" faces of the vertex figure for the vertex δ_u of $B(n)$. We now make this relationship precise. The constructions and proofs in this section rely only partly on the results of 5.5.3, which they generalize. Instead, in this section, a slightly different perspective on the results of 5.5.3 is offered.

In the context of blocking type polyhedra, we define a vertex figures as an intersection of an appropriate hyperplane with the polyhedron. A face figure is an iterated vertex figure, or, an intersection of an appropriate affine subspace with the polyhedron. Combinatorially, the face lattice of a face figure for a face G is a sub-lattice of the face lattice of the polyhedron consisting of all faces which contain G. For $U \subseteq V_n$ we denote by $B(n)/\bar{U}$ the face figure of $B(n)$ for the face *$B(n)/\bar{U}$*

$$\bar{U} := \bigvee_{v \notin U} \delta_v.$$

We will identify the faces of the face figure $B(n)/\bar{U}$ with the corresponding faces of the blocking polyhedron $B(n)$. We denote by *$\varkappa$*

$$\varkappa_U : \mathscr{L}(B(n)) \to \mathscr{L}(B(n)/\bar{U})$$

the mapping which assigns to every face X of $B(n)$ the face of the face figure $B(n)/\bar{U}$, which corresponds to the face $X \vee \bigvee_{v \notin U} \delta_v$ of $B(n)$. We omit the index U on $\varkappa$ if this does not lead to confusion.

5.8.1 Definition Let $U \subseteq V_n$. The complex which is defined as the set of all intersections of members of $\mathcal{T}_u(F)$ for $u \in U$, is called the *local tilting complex for* U, and is denoted by $\mathcal{T}_U(F)$. *local tilting complex $\mathcal{T}_U$*

A face of GTSP(n) (resp. of $B(n)$) is said to be of *TT-type at u,* if it is not contained in a non-negativity facet (resp. it is bounded) and it is not contained in the degree facet (resp. it does not contain the degree vertex) for node u. *TT-type at u*

We denote the mapping $P \mapsto \varphi(P)$, which associates a member of the face lattice of $B(n)$ to every member of the face lattice of $\mathcal{T}(F)$ by $\varphi(\square)$. Let $\mathscr{F}(\mathcal{T}_U)$ denote the set of facets of the polytopal complex. The following lemma is a generalization of Theorem 5.2.12 or Proposition 5.3.15.

5.8.2 Lemma *Let X be a TT-type face of $B(n)$, and denote $F := X^\sharp \cap STSP(n)$. If* $\dim X = \operatorname{codim} F - 1$*, then $\varkappa_U(X)$ is a face of dimension* $\dim X$ *in $B(n)/\bar{U}$.* □

We define the mapping $s\colon \mathcal{T}(F) \to \mathcal{T}_U(F)$, which assigns to each face P of $\mathcal{T}(F)$ the smallest face of $\mathcal{T}_U(F)$ with contains P, i.e., $s(P) = \min_{Q \in \mathcal{T}, Q \supseteq P} Q$. *$s\colon \mathcal{T} \to \mathcal{T}_U$*

5.8.3 Lemma *There exists a unique mapping $\psi\colon \mathcal{T}_U(F) \to \mathscr{L}(B(n)/\bar{U})$ such that the following diagram commutes*

$$\begin{tikzcd}
\mathcal{T}_U(F) \arrow[r, dotted, "\psi"] & \mathscr{L}(B(n)/\bar{U}) \\
\mathcal{T}(F) \arrow[u, "s"] \arrow[r, hook, "\varphi(\square)"'] & \mathscr{L}(B(n)). \arrow[u, "\varkappa"']
\end{tikzcd} \tag{5.7}$$

The mapping ψ is an injective morphism of the posets. Moreover, it maps the facets of $\mathcal{T}_U(F)$ to the faces of dimension $\operatorname{codim} F - 1$ *of $B(n)/\bar{U}$.*

Proof. Let P, P' be faces of $\mathfrak{T}$ such that $\varkappa(\varphi(P)) = \varkappa(\varphi(\square)(P'))$, and let $a \in \operatorname{relint} P$, $a' \in \operatorname{relint} P'$. Since $\varphi(P) \vee \bigvee_{v \notin U} \delta_v = \varphi(P') \vee \bigvee_{v \notin U} \delta_v$, we know that for all $u \in U$

$$\Delta_u(\varphi(a)) = \Delta_u(\varphi(P)^\sharp) = \Delta_u(\varphi(P')^\sharp) = \Delta_u(\varphi(a')).$$

But this implies, with the notation of Definition 5.5.9 in 5.5.2, that $E_u(a) = E_u(a')$ for all $u \in U$. Since we also have $\mathscr{F}(a) = \mathscr{F}(a')$, we can conclude, by Lemma 5.5.10-(c), that $s(P) = s(P')$. This shows that the mapping ψ exists. The uniqueness is a trivial consequence of the surjectivity of s.

We show ψ honors the order of the two posets. Suppose that $s(P) \subseteq s(P')$. Then $\Delta_u(\varphi(P)^\sharp) = E_u(a) \supseteq E_u(a') = \Delta_u(\varphi(P')^\sharp)$ for all $u \in U$, and $\varphi(P)^\sharp \cap \mathscr{H}(n) = \mathscr{F}(a) \supseteq \mathscr{F}(a') = \varphi(P')^\sharp \cap \mathscr{H}(n)$. From this we see that

$$\left(\varphi(P) \vee \bigvee_{v \notin U} \delta_v\right)^\sharp \supseteq \left(\varphi(P') \vee \bigvee_{v \notin U} \delta_v\right)^\sharp.$$

Thus, the opposite relation holds for the faces $\psi(s(P))$, $\psi(s(P'))$ of $B(n)/\bar{U}$. A similar argument proves that ψ is injective. Namely, if $s(P) \neq s(P')$, then $\mathscr{F}(a) \neq \mathscr{F}(a')$, or there exists a $u \in U$ such that $E_u(a) \neq E_u(a')$, whence $\psi(P) \neq \psi(P')$.

The statement about the facets of $\mathfrak{T}_U(F)$ follows from the previous lemma. $\square$

We now state the main result of this section.

5.8.4 Theorem *The mapping ψ is a combinatorial isomorphism between $\mathfrak{T}_U(F)$ and the subcomplex of the boundary complex of $B(n)/\bar{U}$ consisting of all faces which are of TT-type at each $u \in U$ and which are contained in $F^\sharp \in B(n)/\bar{U}$.*

Proof. We need to show that the image of ψ is the named subcomplex. Since for every $P \in \mathfrak{T}_U(F)$, there exists a $Q \in \mathfrak{T}(F)$ with $s(Q) = P$ and $\psi(P) = \varkappa(\varphi(Q))$, it is clear that $\psi(P)$ is of TT-type at each $u \in U$, and that it is contained in $F^\sharp$. On the other hand, let $G \in B(n)/\bar{U}$ be a face which is of TT-type at each $u \in U$ and which is contained in $F^\sharp$. Then there exists a TT-type face G' of $B(n)$ which is contained in $F^\sharp$ and such that $\varkappa(G') = G$. There also exists a face P' of $\mathfrak{T}(F)$ with $\varphi(P) = G'$, and hence $\psi(s(Q)) = \varphi(\varkappa(P')) = G$. This completes the proof of the theorem. $\square$

5.8.5 Corollary *The subcomplex of the boundary complex of $B(n)/\bar{U}$ consisting of all faces which are of TT-type at each $u \in U$ and which are contained in $F^\sharp \in B(n)/\bar{U}$ (i.e., the antistar of $\bar{U}$ in the complex of $F^\sharp$) is pure of dimension* $\operatorname{codim} F - 1$ *and it is topologically a closed ball.* $\square$

$\mathfrak{L}_U(a)$ Let $U \subseteq V_n$. For each $a \in F^\diamondsuit$, we define the linear space

$$\mathfrak{L}_U(a) := L_U/(L_U \cap L_{\mathscr{H}}) = (L_{\mathscr{H}} + L_U)/L_{\mathscr{H}},$$

where $L_{\mathscr{H}} := \operatorname{lin}(\operatorname{ham}(a) - \operatorname{ham}(a))$ and $L_U := \operatorname{lin}\{\mathfrak{s}_{u,e} \mid u \in U,\, e \in E_u(a)\}$. (We admit that writing $L_U/(L_U \cap L_{\mathscr{H}}) = (L_{\mathscr{H}} + L_U)/L_{\mathscr{H}}$ is somewhat sloppy.)

5.8.6 Corollary *Let $U \subseteq V_n$. For all $P \in \mathfrak{T}_U(F)$ and $a \in \operatorname{relint} P$, we have*

$$\dim F + |U| + \operatorname{codim} P = \dim \psi(P)^\sharp = \dim \operatorname{ham}(a) + \dim \mathfrak{L}_U(a)$$

and

$$\dim P = m - |V_n \setminus U| - 1 - \dim \operatorname{ham}(a) - \dim \mathfrak{L}_U(a).$$

Proof. Apply Lemma 5.3.9. $\square$

5.8.1 An application

We give an application of Theorem 5.8.4. We repeat a definition from [NR93].

5.8.7 Definition Let F be a metric face of GTSP(n) and $u \in V_n$. We define an adjacency relation on $\Delta_u(F)$ by saying that $e \sim f$ iff there exists a point $x \in F \cap \mathscr{E}^1(n)$ such that both $x + \mathfrak{s}_{u,e}$ and $x + \mathfrak{s}_{u,f}$ are in GTSP(n). The resulting graph is denoted by $\mathcal{G}_u^\Delta(F)$.

The following fact is easily seen by induction on the distance of the edges in $\mathcal{G}_u^\Delta(G)$. Actually the proof is the same as that of Lemma 2.14 in [NR93].[4]

5.8.8 Lemma *If e and f are adjacent in $\mathcal{G}_u^\Delta(G)$ then their corresponding shortcuts are equal modulo $G \cap \mathscr{H}(n)$.* □

We see here that the definition of the graph is too weak to reveal much about the relationship between STSP(n) and GTSP(n). However, it can be used as a technical tool to aid bounding the number of different shortcuts modulo $F \cap \mathscr{H}(n)$.

5.8.9 Proposition *Let G be a non-NR facet of GTSP(n) and $F := G \cap$ STSP(n). Let $u \in V_n$ and denote by p the number of facets of $\mathcal{T}_u(F)$ which contain the point of $F^\diamond$ corresponding to G. Further let q denote the maximum number of G-feasible shortcuts at u which are linear independent modulo $G \cap \mathscr{H}(n)$ and let c denote the number of connected components of $\mathcal{G}_u^\Delta(G)$. Then $p \leq q \leq c$ holds.*

Proof. The first inequality follows from Corollary 5.8.6, the second from the lemma above. □

We apply this proposition and Theorem 5.8.4 to 0-node lifting. For simplicity we repeat the definition of 0-node lifting for a special case (it is essentially equivalent to the definition in 2.1.1). Let $c \in \mathbb{R}^{E_n}$. We can identify c with a symmetric $n \times n$-matrix C with zeros in the diagonal. If we duplicate the $(n-1)$th row and column of this matrix, we obtain a symmetric $(n+1) \times (n+1)$-matrix C° with zeros in the diagonal, which can be identified with a vector $c^\circ \in \mathbb{R}^{E_{n+1}}$. We say that c° is obtained by *0-node lifting* of c. If c is a TT-type vertex of $B(n)$, then c° is a TT-type vertex of $B(n+1)$ [NR91].

c°

0-node lifting

Queyranne & Wang [QW93] proved a result on 0-node lifting for the STSP. In our terminology, it states that if c is an NR-vertex, and $c^{\circ\circ}$ is obtained by 0-node lifting c twice, then the vertex $c^{\circ\circ}$ of $B(n+2)$ is an NR-vertex. We extend their result to the following theorem. Note that, while the statement [QW93] is only about $c^{\circ\circ}$ itself, the statement of the following theorem is about all non-NR facets which are in the same "area" of $B(n)$, i.e., in the image under the mapping φ of the tilting complex.

5.8.10 Theorem *Let c be a TT-type vertex of $B(n)$ and $c^{\circ\circ}$ be obtained from c by 0-node lifting c twice. If $c^{\circ\circ}$ is non-NR vertex and $(c^{\circ\circ}, 1)$ defines the non-NR facet G, then, with $F := G \cap$ STSP(n), the local tilting complexes $\mathcal{T}_u(F)$ at $u = n-1, n, n+1$ are trivial, i.e., they are equal to the complex of the polytope $F^\diamond$.*

Proof. It is easy to see that the graph $\mathcal{G}_u^\Delta(G)$ is connected. Let $a \in F^\diamond$ with $\varphi(a) = c$, and $u \in \{n-1, n, n+1\}$. By the proposition, a is contained in only one facet of $\mathcal{T}_u(F)$. Since $a \in \operatorname{relint} F^\diamond$, this implies that $\mathcal{T}_u(F)$ is trivial. □

[4] At this point we might add that it is possible to show that the sufficient condition in Lemma 2.15 of [NR93] is also necessary.

5.9 0-Node lifting

As in the other chapters of the theoretical part of this thesis, we deal with lifting in this chapter, too. We start with a remark which views non-NR facets slightly differently than before.

5.9.1 Remark Let c be a TT-type vertex of $B(n)$. Let $C := c + \operatorname{cone}\{\delta_u - c \mid u \in V_n\}$ denote the cone with apex c which is spanned by the degree vertices δ_u, $u \in V_n$. Further, let P denote the convex hull of all TT-type vertices of $B(n)$ except for c. Then $(c, 1)$ defines a non-NR facet if and only if C intersects P. To see this, let A denote the matrix whose columns are the TT-type vertices of $B(n)$, except for c. The inequality $(c, 1)$ defines a non-NR facet if and only if there exists a vector $\mu \geq 0$ with $\mathbf{1}\mu = 1$ and a vector $\lambda^0 \geq 0$ such that

$$(1 - \mathbf{1}\lambda^0)\begin{pmatrix} c \\ 1 \end{pmatrix} = \begin{pmatrix} A \\ \mathbf{1}^\top \end{pmatrix}\mu - \begin{pmatrix} D \\ \mathbf{1}^\top \end{pmatrix}\lambda^0.$$

This can be rewritten as

$$c + \sum_{u \in V_n} \lambda^0_u \cdot (\delta_u - c) \in \operatorname{conv} A.$$

From this remark we see again that, if c is not adjacent to a degree vertex δ_u, then it must correspond to a non-NR facet, as the half line which starts at c and runs through δ_u intersects the face $c \vee \delta_u$ of $B(n)$, which contains other TT-type vertices of $B(n)$. The question remains open, whether there is a non-NR-facet which is adjacent to all degree facets. In all examples of non-NR facets which we found, there were non-adjacent degree facets.

Now we introduce our result.

5.9.2 Theorem *Let c be an NR-vertex of $B(n)$ and let c° be the vertex of $B(n+1)$ obtained from c by 0-node lifting. Then for every $U \subseteq V_n$ with $|U \cap \{n-1, n\}| \leq 1$, the face $c^\circ \vee \bigvee_{u \in U} \delta_u$ of $B(n+1)$ is a $|U|$-simplex.*

$\hat{B}(n)$, $\hat{B}_0(n+1)$

For the proof of the theorem, we are happy to be able to introduce a geometrically intuitive lemma. We denote by $\hat{B}(n)$ the complex of all bounded faces of $B(n)$, and by $\hat{B}_0(n+1)$ the complex of all bounded faces of $B(n+1)$ which are contained in the non-negativity facet for the edge $\{n-1, n\}$.

G°

5.9.3 Lemma *The mapping $\Box^\circ \colon G \mapsto G^\circ := \{a^\circ \mid a \in G\}$ induces an affine isomorphism of the complex $\hat{B}(n)$ to the complex $\hat{B}_0(n+1)$.*

Proof. Clearly, $\Box^\circ$ is a bijection of the vertices of the two complexes. It remains to be shown that for a set of vertices M of $B(n)$, M is the set of vertices of a bounded face of $B(n)$ if and only if $\Box^\circ(M)$ is the set of vertices of a bounded face of $B(n+1)$. This can be done by some technical arguments which we omit. $\Box$

5.9.4 Lemma *If $(c^\circ, 1)$ defines a non-NR facet G of GTSP$(n+1)$ which is 0-node lifted, i.e., $c^\circ_{\{n-1,n\}} = 0$, then, with $F := G \cap$ STSP$(n+1)$, for all $P \in \mathcal{T}(F)$*

$$\dim \mathcal{L}_{\{n-1,n\}}(a) = \dim \mathcal{L}_{n-1}(a) + 1 = \dim \mathcal{L}_n(a) + 1.$$

Proof. We show the first equation. By Lemma 5.3.11, we have $\dim \mathcal{L}_{\{n-1,n\}}(a) \geq \dim \mathcal{L}_{n-1}(a) + 1$. For the reverse inequality, we prove the following two claims.

Claim 1. If $e \in \Delta_n(G)$ then there exists $f \in \Delta_n(G) \cap \partial(n-1)$ with $\mathfrak{s}_{n,e} = \mathfrak{s}_{n,f}$ in $\mathcal{L}_{\{n-1,n\}}(a)$.

Claim 2. If $e, f \in \Delta_n(G)$, then $\mathfrak{s}_{n,e} - \mathfrak{s}_{n,f} \in \mathcal{L}_{n-1}$.

From the second claim, it follows that the vector space $\mathcal{L}_{\{n-1,n\}}(a)/\mathcal{L}_{n-1}(a)$ has dimension at most one, and hence the dimension differs by at most one.

Proof of Claim 1. Let $uv = e \in \Delta_n(G) \setminus \partial(n-1)$. It is easy to see that there exists an $x \in \mathscr{C}^1(n+1)$ and an f for which $x + \mathfrak{s}_{n,e}, x + \mathfrak{s}_{n,f} \in G \cap \mathscr{H}(n+1)$. This proves the claim.

Proof of Claim 2. By Claim 1, we only need to show that Claim 2 holds for $e, f \in \Delta_n(G) \cap \partial(n-1)$. With $e = \{u, n-1\}$ and $f = \{u', n-1\}$, and $g := \{n-1, n\}$, we compute

$$\mathfrak{s}_{n,e} - \mathfrak{s}_{n,f} = \big(\chi^e - \chi^{un} - \chi^g\big) - \big(\chi^f - \chi^{u'n} - \chi^g\big) = \mathfrak{s}_{n-1,u'n} - \mathfrak{s}_{n-1,un}.$$

Since $u'n, un \in \Delta_{n-1}(G)$, the statement of the claim is true. □

We extend the terminology defined in Section 5.8 a bit. For a face G of $B(n)$ which is not an intersection of non-negativity facets, and $U \subseteq V_n$, we let $L_{\mathscr{H}}(G)$ denote the linear space defined by $\mathrm{STSP}(n) \cap G^\sharp$, and denote by $L_U(G)$ the linear space generated by the set of feasible shortcuts for $G^\sharp$. Then, we define

$L_{\mathscr{H}}(G)$, $L_U(G)$, $\mathcal{L}_U(G)$

$$\mathcal{L}_U(G) := L_U(G) \;/\; \big(L_U(G) \cap L_{\mathscr{H}}(G)\big) = \big(L_{\mathscr{H}}(G) + L_U(G)\big) \;/\; L_{\mathscr{H}}(G).$$

5.9.5 Lemma *Let X be a bounded face of $B(n)$. For all $U \subseteq V_n$ with $n-1 \in U$, we have $\dim \mathcal{L}_U(X) - \dim \mathcal{L}_{n-1}(X) \geq \dim \mathcal{L}_U(X^\circ) - \dim \mathcal{L}_{n-1}(X^\circ)$.*

Proof. We abbreviate $L_U := L_U(X)$ and $L_U^\circ := L_U(X^\circ)$, and the same for U replaced by $\mathscr{H}$ or $n-1$. We construct a surjective linear mapping $\mathcal{L}_U/\mathcal{L}_{n-1} \to \mathcal{L}_U^\circ/\mathcal{L}_{n-1}^\circ$. By elementary algebra, this means finding a surjective mapping f such that the following diagram commutes:

$$\begin{tikzcd}
(L_{\mathscr{H}} + L_U) \;/\; (L_{\mathscr{H}} + L_{n-1}) \arrow[r, dotted, "f"] & (L_{\mathscr{H}}^\circ + L_U^\circ) \;/\; (L_{\mathscr{H}}^\circ + L_{n-1}^\circ) \\
L_U \arrow[u] \arrow[r, hook, "\imath"'] & L_U^\circ, \arrow[u, "\pi"']
\end{tikzcd}$$

where $\imath\colon \mathbb{R}^{E_n} \hookrightarrow \mathbb{R}^{E_{n+1}}$ maps according to the inclusion $E_n \subseteq E_{n+1}$, and the vertical arrows designate the canonical projections. Now it is possible to show that $\imath$ maps sums of Hamiltonian cycles $\sum_r \xi_r x^r$ with $x^r \in \mathscr{H}(n)$ and $\sum_r \xi_r = 0$ to sums of Hamiltonian cycles in $\mathscr{H}(n+1)$ and shortcuts in L_{n-1}°. Hence, a linear mapping f exists. To show that f is surjective, let $s \in L_U^\circ \setminus \imath(L_U)$. We distinguish three cases.

1. $s = \mathfrak{s}_{n-1,vn}$ for $v \in V_n \setminus \{n-1\}$,
2. $s = \mathfrak{s}_{u,vn}$ for $u \in U \setminus \{n-1\}$, $v \in V_n \setminus \{n-1\}$, and
3. $s = \mathfrak{s}_{u,\{n-1,n\}}$.

In the first case, we have $\pi(s) = 0 \in \operatorname{Im} f$. In the second case, we can write $s = \mathfrak{s}_{u,\{n-1,v\}} + \mathfrak{s}_{n-1,vn} - \mathfrak{s}_{n-1,un}$, whence $\pi(s) \in \operatorname{Im} f$. But the third case, it is possible to show that s is equal, modulo $L_{\mathscr{H}}^\circ$, to a shortcut treated in one of the other cases. □

Now we have all the ingredients to prove the theorem of this section.

Proof of Theorem 5.9.2. We prove that $c^\circ \vee \bigvee_{u \in V_{n+1} \setminus \{n-1\}} \delta_u$ is a simplex in $B(n+1)$. Let $F := G \cap \mathrm{STSP}(n+1)$, and $a \in \operatorname{relint} F^\diamondsuit$ with $\varphi(a) = c^\circ$. With $\gamma := \dim F^\diamondsuit$, and using Lemma 5.9.5, we compute

$$\begin{aligned}
n-1 \geq{} & \dim \mathcal{L}_{V_n}(c) - \dim \mathcal{L}_{n-1}(c) \\
& \geq \dim \mathcal{L}_{V_n}(c^\circ) - \dim \mathcal{L}_{n-1}(c^\circ) \underset{(*)}{=} n + 1 + \gamma - 1 - \dim \mathcal{L}_{n-1}(c^\circ),
\end{aligned}$$

where the equation $(*)$ follows from Lemma 5.9.4. Hence, $\dim \mathcal{L}_{n-1}(c^\circ) \geq \gamma + 1$. By Corollary 5.8.6, if P is the face of $\mathcal{T}_{n-1}(F)$ containing a as a relative interior point, we have

$$0 \leq \dim P \leq \binom{n+1}{2} - n - 1 - \dim F - (\gamma + 1) = 0.$$

This implies that $P = \{a\}$ is a vertex. Invoking Theorem 5.8.4 with $U := V_{n+1} \setminus \{n-1\}$, we see that the image of a is a vertex of the face figure, which implies the statement of the theorem by a standard uniqueness argument in the style of Theorem 5.2.12. $\square$

Chapter 6

Understanding LP-solutions

In this chapter we aim to understand which properties of a vertex x of a relaxation of $\mathrm{GRP}(\Gamma, b)$ are not characteristic for the fact of being outside the polyhedron. We will identify structures which can be changed or replaced without changing the fact that $x \notin \mathrm{GRP}(\Gamma, b)$. This issue has been treated in the context of so-called "safe shrinking" operations, where the results are applied to vertex LP-solutions in a Branch-and-Cut-algorithm to reduce the size of the *support graph* $G(x)$. For the STSP, this has been studied in the paper by Padberg & Rinaldi [PR90a].

We introduce a collection of modifications which can be performed on x without changing the property $x \notin \mathrm{GRP}(\Gamma, b)$. Although some of these modifications have been implemented (see, e.g., 10.1.1 or Section 10.5), our aim is mainly to increase the understanding of what makes an LP-solution "broken", i.e., what causes $x \notin \mathrm{GRP}(\Gamma, b)$.

6.0.1 Terminology

In this chapter we will deal with tuples (Γ, b, x) consisting of a GRP-structure $\Gamma = (G, \mathcal{C}, \mathbf{t})$, a vector of upper bounds $b \colon E(G) \to \mathbb{Z}_+^*$, and a vector $x \in \mathbb{R}^{E(G)}$ satisfying $x \leq b$. We study *modification operations* $(\Gamma, b, x) \mapsto (\Gamma', b', x')$, which are *safe*, i.e., $x \notin \mathrm{GRP}(\Gamma, b)$ if and only if $x' \notin \mathrm{GRP}(\Gamma', b')$. In the opposite direction, we will give operations for *pulling back* valid inequalities separating x' from $\mathrm{GRP}(\Gamma', b')$ to valid inequalities separating x from $\mathrm{GRP}(\Gamma, b)$. Sometimes an operation will offer a set of possible modifications. In that case, we speak of a *safe alternative operation,* if $x \notin \mathrm{GRP}(\Gamma, b)$ holds if and only if there exists an i such that $x_i \notin \mathrm{GRP}(\Gamma_i, b_i)$, and we use the symbolic notation $(\Gamma, b, x) \mapsto \{(\Gamma_1', b_1', x_1'), \dots, (\Gamma_k', b_k', x_k')\}$.

safe operation

pulling back

safe alternative operation

In this chapter, unless otherwise stated, all graphs are simple, and the node identification operations produce simple graphs. In this process, vectors x or b are adopted as described in 0.2.2. We abbreviate $G^{\mathrm{s}}_{\mathcal{C}}(x/E_{\mathrm{int}})$ to $G_{\mathcal{C}}(x)$.

6.1 Blocks and connected components

We start with the the following fact. Suppose that $G_{\mathcal{C}}(x)$ is connected. Let F be a set of R-internal edges such that in the graph $G(x) \cup F$ the subgraph induced by each R-set is connected. Further, let $G_1, \dots, G_b$ be a block decomposition of $G(x)$. For $j = 1, \dots, b$ define a parity function $\mathbf{t}_j$ on G_j by shrinking the other blocks into their respective cut-nodes in G_j, and define a partition $\mathcal{C}_j$ into R-sets by restricting $\mathcal{C}$ to the node set of G_j. Let $\Gamma_j := (G_j, \mathcal{C}_j, \mathbf{t}_j)$.

6.1.1 Proposition *With these notations, $x \notin \mathrm{GRP}(\Gamma, b)$ if and only if there exists a $j \in \{1, \dots, b\}$ such that $x|_{E(G_j)} \notin \mathrm{GRP}(\Gamma_j, b|_{E(G_j)})$.* □

Inequalities on a block G_j are pulled back by 0-node lifting, where each cut-node u contained in G_j is replaced by the set of nodes of G contained in blocks which are connected to G via u. It is known from [CLS01] that this reduction is safe both for the class of PBs and 2-PBs, which means that, if, for $\mathrm{GRP}(\varGamma, b)$, there exists a violated inequality in such a class, then, for one of the blocks, there still exists a violated inequality in the same class.

A second simple modification allows to decompose into connected components under certain conditions. Let H be a connected component of $G(x)$, such that $V(H) \subseteq C$ for an R-set $C \in \mathcal{C}$. Let $G' := G - V(H)$, and denote by $\mathcal{C}'$ the R-set partition with C replaced by $C \setminus V(H)$. Let $\varGamma' := (G', \mathcal{C}', \mathbf{t}|_{V(G)\setminus V(H)})$.

6.1.2 Proposition *With the above notations, $x \notin \mathrm{GRP}(\varGamma, b)$ if and only if x violates a blossom inequality* (2.5a) *with $U \subseteq V(H)$ or $x|_{E(G')} \notin \mathrm{GRP}(\varGamma', b|_{E(G')})$.*

Proof. The conditions are clearly sufficient for $x \notin \mathrm{GRP}(\varGamma, b)$. Suppose that no such blossom inequality is violated; in particular $V(H)$ is an even set. Then $x|_{E(H)}$ is a convex combination of T-joins:

$$x|_{E(H)} = \sum_k \lambda_k y^k.$$

If $x|_{E(G')} \in \mathrm{GRP}(\varGamma', b|_{E(G')})$, then $x|_{E(G')}$ is a convex combination of semitours in $\mathscr{S}(\varGamma', b|_{E(G')})$:

$$x|_{E(G')} = \sum_l \mu_l z^l.$$

For every pair k, l, we have $(y^k, \mathbf{0}, z^l)^\top \in \mathscr{S}(\varGamma)$. Hence,

$$x = \sum_{k,l} \lambda_k \mu_l \begin{pmatrix} y^k \\ \mathbf{0} \\ z^l \end{pmatrix},$$

which means that x is a convex combination of semitours in $\mathscr{S}(\varGamma, b)$, and the proof is completed. □

6.2 Flipping variables and merging R-sets

Let $f = uv \in E(G)$ satisfy $b_f < \infty$, and suppose that u and v are contained in the same set $C \in \mathcal{C}$. Let $\varGamma_{\chi^{\{u,v\}}}$ as defined in Section 3.1 on page 25, and define $x' := \mathrm{flip}^{[b;\chi^f]}(x)$. See Section 3.1 for the definitions of $\mathrm{flip}^{[b;\cdot]}$, $a'^{[\chi^f]}$, and $\alpha' - (b \odot \chi^f)a'$.

6.2.1 Proposition *The operation $(\varGamma, b, x) \mapsto (\varGamma_{\chi^{\{u,v\}}}, b, x')$ is safe. If (a', α') is a valid inequality for $\mathrm{GRP}(\varGamma_{\chi^{\{u,v\}}}, b)$, then a pulled back inequality (with the same amount of violation) is defined by $\big(a'^{[\chi^f]}, \alpha' - (b \odot \chi^f)a'\big)$.*

Proof. This is an easy consequence of Propositions 3.1.2 and 3.1.5. □

6.2.2 Corollary *Let $F := \{f = uv \in E_{\mathrm{int}}(\varGamma) \mid x_f = b_f\}$. If the parities $\mathbf{t}$ are modified accordingly, then the set of edges F can safely be removed from the support graph.* □

To understand the benefit of the next propositions, let the reader be reminded of the fact that the GRP is polynomial if the number of R-sets is bounded by a fixed constant, see Section 1.2. By the polynomial equivalence of separation and optimization [GLS93], we could heuristically argue that the separation problem becomes computationally easier if the number of R-sets is reduced. Let $f = uv \in E(G)$ be an R-external edge with $b_f < \infty$ and $x_f = b_f$. Recall the definition of $\mathcal{C} \circledcirc f$ from Section 3.2, namely if $u \in C_u \in \mathcal{C}$, $v \in C_v \in \mathcal{C}$, then $\mathcal{C} \circledcirc f := \mathcal{C} \setminus \{C_u, C_v\} \cup \{C_u \cup C_v\}$. Let $\varGamma' := (G, \mathcal{C} \circledcirc f, \mathbf{t})$.

6.2.3 Proposition *If $x_f = b_f$, then the operation $(\varGamma, b, x) \mapsto (\varGamma', b, x)$ is safe, i.e., the two R-sets can safely be "merged". Violated inequalities can be pulled back without changing them.*

Note that the vector x is not changed.

Proof. The fact that $x \in \mathrm{GRP}(\mathit{\Gamma}, b)$ implies $x \in \mathrm{GRP}(\mathit{\Gamma}', b')$ is a consequence of Remark 3.2.2 on page 27.

It remains to show that if $x \notin \mathrm{GRP}(\mathit{\Gamma}, b)$ then we also have $x \notin \mathrm{GRP}(\mathit{\Gamma}', b')$. Suppose that x is a convex combination of semitours $z^1, \ldots, z^r \in \mathscr{S}(\mathit{\Gamma}', b)$, i.e.,

$$x = \sum_j \lambda_j z^j$$

Since $x_f = b_f < \infty$ we have $z^j_f = b_f$ for all j. But this implies that the z^j connect all the sets $C \in \mathcal{C}$, i.e., $z^j \in \mathscr{S}(\mathit{\Gamma}, b)$. □

By combining Propositions 6.2.3 and 6.2.1, we obtain the following corollary.

6.2.4 Corollary *If the partition $\mathcal{C}$ and the parities $\mathbf{t}$ are modified accordingly, then all edges f for which the upper bound inequality $x_f \le b_f$ is satisfied with equality, can safely be removed from the support graph $G(x)$.* □

The condition $b_f < \infty$ can be avoided by requiring that certain connectivity inequalities must be tight. Let $f = uv$ be an R-external edge and let S_u, S_v be unions of R-sets with $u \in S_u$ and $v \in S_v$. The proof of the following proposition is a mixture of the proof of Propositions 6.2.3 and 6.2.6 below.

6.2.5 Proposition *Suppose that $x_f = 1$ and $x(\partial(S_u)) = x(\partial(S_v)) = 2$. If there is at least one R-set which is not contained in either S_u or S_v, and the connectivity inequality for the union of R-sets $S_u \cup S_v$ is not violated, then $\mathcal{C}$ can be replaced by $\mathcal{C} \circledcirc f$.* □

Under the same conditions, the bound of the edge f can be reduced to one. Define a new vector of upper bounds by $b'_e = b_e$ for all $e \ne f$ and $b'_f = 1$. What is peculiar about this operation is that there is no obvious way to pull back a violated inequality.

6.2.6 Proposition *Under the conditions of Proposition 6.2.5 the operation $(\mathit{\Gamma}, b, x) \mapsto (\mathit{\Gamma}, b', x)$ is safe.*

Proof. Clearly, if $x \in \mathrm{GRP}(\mathit{\Gamma}, b')$ then $x \in \mathrm{GRP}(\mathit{\Gamma}, b)$. On the other hand, suppose that $x \in \mathrm{GRP}(\mathit{\Gamma}, b)$. Let $z^1, \ldots, z^r$ be a family of semitours in $\mathscr{S}(\mathit{\Gamma}, b)$, and $\lambda_1, \ldots, \lambda_r \ge 0$ numbers such that $x = \sum_{j=1}^r \lambda_j z^j$, and $\sum \lambda_j = 1$. Clearly, we have $z^j(\partial(S_u)) = z^j(\partial(S_v)) = 2$ for all j. Since there are R-sets which are not contained in S_u or S_v, we have $x(S_u : S_v) = x_f = 1$. From this it follows that $z^j_f = 1$ for all j, because $z^j_f \ge 2$ is impossible. Thus we have $z^j \in \mathrm{GRP}(\mathit{\Gamma}, b')$, which implies $x \in \mathrm{GRP}(\mathit{\Gamma}, b')$. □

The last operation in this section is based on circular partitions. Let k be an integer ≥ 1 and let $U_0, \ldots, U_{k+1}$ be a partition of the node set of G into unions of R-sets. Suppose that $U_j \in \mathcal{C}$ for $j = 1, \ldots, k$, i.e., $U_1, \ldots, U_k$ are single R-sets.

6.2.7 Proposition *If*

$$x(\partial(U_j)) = 2 \textit{ for } j = 0, \ldots, k+1, \textit{ and} \tag{$*$}$$

$$x(U_j : U_{j+1}) = 1 \textit{ for } j = 0, \ldots, k, \textit{ and } x(U_{k+1} : U_0) = 1, \tag{$**$}$$

then the R-set $U_1, \ldots, U_k$ can be safely merged to form one single R-set $U_1 \cup \cdots \cup U_k$.

Proof. Denote by $\mathcal{C}'$ the new set of R-sets. Let z be a semitour for $\mathcal{C}'$ which is in a family whose convex sum equals x. We show that z is also a semitour for $\mathcal{C}$. This can be done by a parity argument. For each of the cuts $F_j := \partial(U_0 \cup \dots \cup U_j)$ for $j = 0, \dots, k$ the value $z(F_j)$ must be an even number. Because of $(*)$ above,

$$z(\partial(U_0)) = z(\partial(U_{k+1}) = z(\partial(U_1 \cup \dots \cup U_k)) = 2$$

must hold, so that $z(U_0 : U_{k+1}) = 1$. But then $z(F_j) \geq 2$, which implies $z(U_j : U_{j+1}) \geq 1$, because $x(U_i : U_l) = 0$ for $i \in \{0, \dots, j\}$ and $l \notin \{0, \dots, j\}$ if $(i, j) \notin \{(0, k+1), (j, j+1)\}$. Thus z connects all the (original) R-sets U_j, $j = 1, \dots, k$, to their respective predecessors and successors. □

6.3 Shrinking

In this section we will deal with safe shrinking operations. For the Symmetric TSP, conditions under which shrinking operations are safe are discussed by Padberg & Rinaldi [PR90a]. Up to now, no comparable results are known for the GTSP, or the GRP.

For the remainder of the section, we will assume that x violates no connectivity inequality.

6.3.1 Shrinking based on tightness of connectivity inequalities

We start by showing how, under similar conditions as those in [PR90a], we can obtain safe shrinking operations for the GRP. We follow [PR90a] in speaking of "shrinking", instead of node identification, although in the case of the GRP, the subgraphs which we identify to single nodes need not be connected.

For a union of R-sets U, we define the GRP-structures Γ/U by identifying the node set U to a new node, in the sense of node identification in simple graphs. The R-set partition is changed as in Definition 2.1.5, the parity of the new node is the parity of the set U. The bound vector $b \colon E(G) \to \mathbb{Z}_+^*$ is treated as described in 0.2.2; the bound vector on the graph G/U is denoted by b/U. The same applies to the vector x.

Now, let a partition of the node set of $V(G)$ into three sets $U, W, \{v\}$ be given, such that v is an R-isolated node, and U and W are unions of R-sets. The proof of the following proposition is almost the same as that for Theorem 6.7 in [PR90a]. We do not repeat it here.

6.3.1 Proposition *If $x(v : U) = x(U : W) = x(W : v) = 1$, then we have the following operation: at least one of U or W can safely be identified to a single node, i.e., we have the safe alternative operation $(\Gamma, b, x) \mapsto \{(\Gamma/U, b/U, x/U), (\Gamma/W, b/W, x/W)\}$.* □

Note that after each of the alternatives, Proposition 6.2.5 can be applied, so that $\{v, u\}$ (or $\{v, w\}$ respectively) form a new R-set.

Following [PR90a], we call a path $u_0, e_1, u_1, \dots, e_k, u_{k+1}$ with $x_{e_i} = 1$ for all i, and such that the u_i for $i = 1, \dots, k$ are R-isolated (required) nodes with $x(\partial(u_i)) = 2$ a *1-path.*

6.3.2 Corollary *Let $u_0, e_1, u_1, \dots, e_k, u_k$ be a 1-path, and suppose*

$$x(\partial(u_0)) = x(\partial(u_{k+1})) = 2$$

holds. If u_0 and u_{k+1} are R-isolated, then it is safe to shrink the 1-path to a single edge linking u_0 to u_{k+1}.

Proof. Define $U := \{u_1, \dots, u_{k+1}\}$ and $W := V(G) \setminus \{u_0, \dots, u_{k+1}\}$. Then Proposition 6.3.1 can be applied with $v := u_0$. But if we shrink the set W, we get a semitour, so that if $x \notin \mathrm{GRP}(\Gamma, b)$, we must have $x/U \notin \mathrm{GRP}(\Gamma/U, b/U)$. □

Now we give a new shrinking condition, tailored for the GRP. Let a partition of the node set of $V(G)$ into two sets U, W be given, such that U and W are unions of R-sets, and let $f = u_0w_0 \in (U : W)$, where $u_0 \in U$ and $w_0 \in W$. Define the GRP-structure $\mathit{\Gamma}_W$ on the graph G/U. The nodes resulting from identifying U to a single node is again denoted by u_0. The R-set partition shall be $\mathcal{C}_W := \mathcal{C}/E(U) \circledcirc f$ and the parities are defined by $\mathbf{t}_W(v) = \mathbf{t}(v)$ if $v \in W \setminus \{w_0\}$, $\mathbf{t}_W(w_0) = 1 - \mathbf{t}(w_0)$ and $\mathbf{t}_W(u_0) = 1$. Further, we let $(x_W)_e := (x/U)_e$ for all $e \neq uw$ and $(x_W)_{vw} := 0$. The GRP-structure $\mathit{\Gamma}_U$ and vector x_U are defined in the same way, and when w_0 denotes the node of G/W which results from the identification of W, $\mathbf{t}_U(w_0) = 1$ and $\mathbf{t}_U(u_0) = 1 - \mathbf{t}(u_0)$.

6.3.3 Proposition *Suppose that $x(u_0 : W) = 1 = x(U : w_0)$; in other words, the neighbors of u_0 and w_0 in $G(x)$, except w_0 and u_0, are all on the same side of the cut $(U : W)$. If $x_f = b_f = 1$ and $x(U : W) = 2$ hold, then the alternative operation $(\mathit{\Gamma}, b, x) \to \{(\mathit{\Gamma}_W, b/U, x_W), (\mathit{\Gamma}_U, b/W, x_U)\}$ is safe.*

Proof. Let $Y \subseteq \mathscr{S}(\mathit{\Gamma}_U, b/W)$ and $Z \subseteq \mathscr{S}(\mathit{\Gamma}_W, b/U)$ such that we have the following convex combinations:

$$x_U = \sum_{y \in Y} \lambda_y y \qquad \text{and} \qquad x_W = \sum_{z \in Z} \mu_z z.$$

Every $y \in Y$ satisfies the R-odd cut inequality $y(U) \geq 1$ with equation, hence there exists a unique $u \in U$ with $y_{uw_0} \neq 0$. For $u \in U$, let $\Xi(u)$ be the set of $y \in Y$ with $y_{uw_0} \neq 0$. The sets $\Xi(u)$, $u \in U$, are pairwise disjoint and their union is Y. Similarly, for $w \in W$, let $\Xi(w)$ be the set of all $z \in Z$ with $z_{u_0w} \neq 0$. Again, The sets $\Xi(w)$, $w \in W$, are pairwise disjoint and their union is Z. If $u \in U$, $w \in W$ with $uw \in E(G)$, and $y \in \Xi(u)$, $z \in \Xi(w)$, then we can define a semitour $y \bowtie z \in \mathscr{S}(\mathit{\Gamma}, b)$ by letting, for all $e \in E(G)$,

$$(y \bowtie z)_e = \begin{cases} y_e & \text{if } e \in E(U) \\ z_e & \text{if } e \in E(W) \\ 1 & \text{if } e \in \{u_0w_0, uw\}. \end{cases}$$

Now, following [PR90a], for $u \in U$ and $w \in W$ with $uw \in E(G)$, and $y \in \Xi(u)$, $z \in \Xi(w)$, define

$$\nu_{yz}^{uw} := \frac{\lambda_y \mu_z x_{uw}}{x(u : W) x(U : w)}.$$

It is easy to see that

$$x = \sum_{\substack{u \in U, w \in W \\ uw \in E(G) \\ y \in \Xi(u) \\ z \in \Xi(w)}} \nu_{yz}^{uw} \cdot (y \bowtie z)$$

is a convex combination. □

Applying this proposition, we obtain the following corollary.

6.3.4 Corollary *Let a partition of the node set of $V(G)$ into two sets U, W be given, such that U and W split precisely one R-set C, i.e. $C \cap U \neq \emptyset$ and $C \cap W \neq \emptyset$ but $C' \subseteq U$ or $C' \subseteq W$ for all $C' \in \mathcal{C}$, $C' \neq C$.*

Under each of the following two conditions, at least one of U or W can be safely shrunk into a node with odd parity which shall be made a member of the R-set C.

1. *$\mathbf{t}(U) = 1 \bmod 2$, $x(U : W) = 1$, and there exist nodes $u \in U \cap C$, $w \in W \cap C$ with $x(u : W) = x(U : w) = 0$.*
2. *$\mathbf{t}(U) = 0 \bmod 2$, $x(U : W) = 2$ and there exists an edge $f = uw \in (C \cap U : C \cap W)$ with $b_f = x_f = 1$, and $x(u : W) = x(U : w) = 1$.*

In the second case, the edge f is deleted, and the parity of its end node in the set which is not shrunk is flipped. □

6.3.2 Shrinking operations involving other inequalities

Not only connectivity inequalities can be used for safe shrinking. We now describe how safe shrinking can be based on R-odd cuts or even more subtle inequalities like KCs.

6.3.5 Proposition *Let $U \subseteq V(G)$ be an odd set which is entirely contained in an R-set, i.e., there exists $C \in \mathcal{C}$ with $C \supseteq U$. Let x satisfy the odd-cut inequality defined by U with equality: $x(U) = 1$. Suppose that there does not exist a violated blossom inequality of the form*

$$x(\partial(W) \setminus F) - x(F) \geq 1 - b(F), \text{ where } \quad W \subseteq U \text{ and } F \subseteq \partial(W) \cap \{e \mid b_e < \infty\}.$$

Then it is safe to identify the set U to a single node.

Proof. The proof is similar to the one of 6.3.1. Let H denote the loopless multigraph which results after identifying the set $W := V(G) \setminus U$ to a single node. By Proposition 2.5a on page 21, $x_{|E(H)}$ is a convex combination of T-joins:

$$x_{|E(H)} = \sum_{y \in Y} \lambda_y y.$$

Now we assume that x/U is also a convex combination of semitours for the shrunk GRP-structure:

$$x/U = \sum_{z \in Z} \mu_z z.$$

Now,for $u \in U$, we define $\Xi(u)$ as the set of all $y \in Y$ with $y_{uw_0} = 1$, where w_0 denotes the node which results from identifying the set W. For $w \in W$, we let $\Xi(w)$ be the set of all $z \in Z$ with $z_{u_0 w} = 1$, where u_0 denotes the node which results from identifying the set U. For $uw \in E(G)$, and $y \in \Xi(u)$, $z \in \Xi(w)$, we define $y \bowtie z \in \mathscr{S}(\Gamma, b)$ by

$$(y \bowtie z)_e = \begin{cases} y_e & \text{if } e \in E(U) \\ z_e & \text{if } e \in E(W) \\ 1 & \text{if } e = uw \end{cases} \qquad \text{for all } e \in E(G),$$

and we let

$$\nu_{yz}^{uw} := \frac{\lambda_y \mu_z x_{uw}}{x(u : W) x(U : w)}.$$

Again, it is easy to see that

$$x = \sum_{u,v,y,z} \nu_{yz}^{uw} \cdot (y \bowtie z)$$

is a convex combination. □

Last not least we give a result for shrinking R-sets to nodes which is safe for GRP(Γ, ∞). The conditions can be checked by a separation routine for KC-inequalities (see page 18) which we propose in 8.3.1.

6.3.6 Proposition *Let $C \in \mathcal{C}$ with $x(\partial(C)) = 2$, let there be node sets A, B with $x(\partial(A)) = x(\partial(B)) = 2$ and nodes $u_0, v_0 \in C$ such that $x(C : A)x(u_0 : A) = x(C : B) = x(v_0 : B) = 1$. If there does not exist a violated R-odd cut inequality $x(\partial(U)) \geq 1$ with $U \subseteq C$ or a violated KC-inequality with $K = 3$, $B_1 = A$, $B_2 = B$, and $B_0 \cup B_K = C$, then it is safe for* GRP(Γ, ∞) *to shrink C to a node.*

Proof. It will become clear in Section 8.3, that, if no such R-odd cut or KC-inequality exists, then $x_{|E(C)}$ lies inside the T-join polyhedron for the graph $G[C]$, where a node $u \in C$ is odd iff $\mathbf{t}(u) + \chi^{\{u_0, v_0\}}$ is odd. If we also assume that x/U is a convex combination of semitours for the shrunk GRP-structure, then it is easy to see that x can be expressed as a convex combination of semitours in $\mathscr{S}(\Gamma, \infty)$. □

Part II

Separation and related algorithms

Chapter 7

Odd cuts and related concepts

In their seminal 1982 paper, Padberg & Rao [PR82] introduced separation algorithms for two kinds of so-called blossom inequalities, namely the capacitated and the uncapacitated cases. In this chapter we improve on both algorithms. The Padberg-Rao algorithm for uncapacitated blossom separation is a generic algorithm to compute a minimum T-odd cut. In this chapter we identify a cut by a pair $(U, V(G) \setminus U)$ where $\emptyset \subsetneq U \subsetneq V(G)$, and for an even cardinality set $T \subseteq V(G)$ a cut $(U, V(G) \setminus U)$ is called *T-odd* or just *odd,* if $|T \cap U| = |T \setminus U| = 0 \bmod 2$, and the minimum T-odd cut problem asks for a T-odd cut whose weight (or capacity) is minimum with respect to a non-negative vector c of edge costs. The algorithm which Padberg & Rao [PR82] propose for this problem produces a so-called Gomory-Hu cut-tree ([GH61]) by a sequence of $|T| - 1$ max-flow computations, and then selects the best odd cut among $|T| - 1$ candidates which are stored in the cut-tree. It has been noted repeatedly that the computational effort required by this procedure is quite high. We improve on this algorithm in the following way: we propose a simple recursive algorithm with the same worst-case running time, i.e., it also solves $|T| - 1$ max-flow problems in the worst case. (We note that the core recursive algorithm was found independently, though earlier, by Rizzi [Riz03].) However, avoiding the computation of a cut-tree allows us to use shrinking operations, which in practice greatly reduce the number of max-flows which are actually computed – and thus the running time. In Section 7.2, we explain the algorithm. Computational results can be found in Section 11.1 in the chapter on the performance of separation algorithms, where we give numbers on the performance of the algorithm when applied to the separation of the R-odd-cut inequalities (2.1).

T-odd cut

Padberg-Rao's algorithm for capacitated blossom separation works by invoking their minimum odd cut algorithm for a so-called split graph, which is created from the graph G by subdividing every edge by the insertion of an extra node. As the running time of this algorithm is quite high, namely $O(m^3 \log n)$ (where $n := |V(G)|$ and $m := |E(G)|$), the attempt to speed it up has attracted some attention: Grötschel & Holland [GH87] were able to reduce the running time to $O(nm^2 \log(n^2/m))$ by an easy idea which is difficult to implement, and Padberg & Rinaldi [PR90a] proposed an $O(n^2 m \log(n^2/m))$ heuristic algorithm. Letchford suggested to us to consider an algorithm which deviates only very slightly from the heuristic proposed in [PR90a], and which has the same worst case running time. We were able to prove that Letchford's algorithm is an exact blossom separation algorithm. An extended abstract of this result appeared in the proceedings of the 10th conference on Integer Programming and Combinatorial Optimization, IPCO [LRT04]. We deal with this algorithm in Section 7.3, where we give results on what we call *blossom minimization.* A blossom is a cut $(U, V(G) \setminus U)$ together with a subset F of the edge set $\partial(U)$ of the cut, such that $|T \cap U| + |F|$ is an odd number. We propose an uncrossing result which proves the correctness of Letchford's algorithm, and we also propose a new, recursive algorithm for blossom minimization.

Applications of uncapacitated and capacitated blossom separation are numerous. The Padberg-Rao [PR82] algorithms are frequently used on separation. The minimum odd cut problem occurs in the separation of various kinds of "odd cut" constraints for routing problems among which are the Mixed Chinese Postman Problem, the Windy Postman Problem, the Mixed Postman Problem, the General Routing Problem, which is a generalization of the Rural Postman Problem, and the Capacitated Arc Routing Problem. The prominent application of the capacitated blossom algorithm is the separation of the 2-matching inequalities of the STSP, and constraints whose separation reduces to this problem also occur in routing problems, among them the cocircuit, and simple 2-PB inequalities mentioned in the previous chapters.

In this chapter, we also deal with the problem of finding all all minimum blossoms and with finding a blossom which is minimum subject to the condition that it separates two given nodes (s, t). We emphasize that all blossom algorithms have the same worst-case running time as their odd-cut counterparts.

7.1 Terminology, notation, and known facts particular to this topic

n, m

odd/even nodes

Let G be a simple undirected graph. In this chapter, we abbreviate $n := |V(G)|$ and $m := |E(G)|$. Let $T \subseteq V(G)$ be a set of even cardinality. The members of T are called *odd* nodes, the nodes in its complement, $V(G) \setminus T$, are called *even.* A set of nodes U is called T*-odd* (resp. T*-even*) if $|U \cap T|$ is an odd (resp. even) number. Hence, a T-odd cut is a cut $(U, V(G) \setminus U)$ such that the sets U and $V(G) \setminus U$ are T-odd, and a minimum T-odd cut is an odd cut which minimizes the submodular function

$$U \mapsto c(U) := \sum_{e \in \partial(U)} c_e, \quad U \subseteq V(G),$$

over all T-odd sets U, where $c \in \mathbb{Q}_+^{E(G)}$ is a vector of (strictly positive) capacities defined on the edges of G. The capacity $c(U)$ of a minimum capacity T-odd cut is denoted by $\lambda^1{}_c(G, T)$, or just $\lambda^1(G)$ if the c and T cannot be confused (the one stands for “odd”). We will just speak of odd or even sets or cuts without mentioning the set T, which will be clear from the context. T-odd cuts are called T-cuts by many authors [GLS93, CCPS98].

An odd minimum (s, t)-cut is a minimum (s, t)-cut which is odd, i.e., a cut $(S, V(G) \setminus S)$ which is odd and satisfies $c(S) = \lambda(s, t)$. A minimum odd (s, t)-cut is a cut which is odd and separates s and t, and which has minimum capacity subject to these conditions.

blossom

In the context of blossoms separation (or minimization), we have two capacities associated with each edge, i.e., we have $c, c' \in \mathbb{Q}_+^{E(G)}$. We say that c_e is the *normal* capacity of the edge e, and that c'_e is the *flipped* capacity. A *blossom* is a cut $(U, V(G) \setminus U)$ together with a subset F of the edge set $\partial(U)$ of the cut, such that $|T \cap U| + |F|$ is an odd number. An (s, t)*-blossom* is a blossom (U, F) such that the cut $(U, V(G) \setminus U)$ separates s and t.

We will have to shrink the graphs we are dealing with. In the context of odd cuts, the capacities are treated in the usual additive manner described in 0.2.2. In the context of blossoms, if in the process of shrinking, the edges $\{e_1, \dots, e_r\}$ are merged to form the edge e of the shrunk graph, we define the normal (flipped) capacity of e to be

$$\min\Big\{ \sum_{i=1}^{r} (1 - \epsilon_i) c_{e_i} + \epsilon_i c'_{e_i} \;\Big|\; \epsilon \in \{0, 1\}^r,\ \textstyle\sum_i \epsilon_i \text{ even (odd)} \Big\},$$

which effectively means picking the smallest weights subject to the condition that the number of flipped weights chosen is even (odd). See also Lemma 7.3.2 below. We write G/st for $G/\{s, t\}$.

7.1.1 Equivalent flow trees and cut-trees

equivalent flow tree

7.1.1 Definition ([Gus90]) An *equivalent flow tree* for a simple graph G with weights $c \in \mathbb{R}_+^E$ and set of terminal nodes $T \subseteq V(G)$ is a tree $\mathcal{T}$ with node set $V(\mathcal{T}) = T$ and edge weights $f_{s,t} = \lambda_c(G, s, t)$ for each $\{s, t\} \in E(\mathcal{T})$.

Algorithm 7.1 Gomory-Hu cut-tree core step

```
Input:
    undirected graph $G$, with edge capacities $c\colon E \to \mathbb{Q}_+$,
    non-empty set of terminal nodes $T \subsetneq V(G)$ ($|T|$ not necessarily even),
    cut-tree $\mathcal{T}$ with terminal node set $T$,
    node $t \notin T$.
Output:
    A Gomory-Hu cut-tree $\mathcal{T}'$ for $G$ with terminal node set $T' := T \cup \{t\}$.
1: Find the supernode $R$ of $\mathcal{T}$ containing $t$ and its representative $r$.
2: Construct the shrunk graph $G_R$.
3: Compute a minimum $(r,t)$-cut in $G_R$, denote it by $(X, \bar{X})$.
4: Construct the two new supernodes $R \cap X$ and $R \cap \bar{X}$ with representatives $r$ and $t$ resp.
5: Construct the tree $\mathcal{T}'$ from $\mathcal{T}$ in the following way. Replace the supernode $R$ of $\mathcal{T}$ by the two
   supernodes $R \cap X$ and $R \cap \bar{X}$ in $\mathcal{T}'$. The two new supernodes are joined by an edge with
   weight $\lambda(G_R, r, t)$. For every edge $S_i R$ of $\mathcal{T}$, add an edge between $S_i$ and $R \cap X$, if $s_i \in X$, or
   between $S_i$ and $R \cap \bar{X}$, if $s_i \in \bar{X}$.
```

cut-tree, supernode

7.1.2 Definition ([GH61]) A *Gomory-Hu cut-tree* for a graph G with capacities c and set of terminal nodes T is a tree $\mathcal{T}$ with edge weights f, whose nodes, called *supernodes,* are non-empty subsets of the node set of G which form a partition of $V(G)$ such that each set contains a unique terminal node, called the *representative* of the supernode. We denote the supernodes by capital letters. Every edge AB of $\mathcal{T}$, would partition the node set of G into two parts if it were removed. We say that the edge *induces* a cut. We require that for each edge AB of $\mathcal{T}$ with weight f_{AB}, the terminal nodes $s \in A$ and $t \in B$ satisfy $f_{AB} = \lambda(G, s, t)$ and that AB induces a minimum (s,t)-cut.

Algorithm 7.1 is an incremental variant of the algorithm of Gomory & Hu [GH61] to compute a cut-tree. The algorithm performs $|T| - 1$ max-flow computations on different graphs, which can be considerably smaller than the graph the algorithms started with. We sketch the proof of correctness of Algorithm 7.1. The underlying basic fact is the following uncrossing idea.

7.1.3 Lemma ([GH61]) *Let $(X, \bar{X})$ be a minimum (x,y)-cut, let $s, t \in X$, and $(S, \bar{S})$ be a minimum (s,t)-cut. Then one of the two cuts $(S \cap X, \bar{S} \cup \bar{X})$, $(S \cup \bar{X}, \bar{S} \cap X)$, is a minimum (s,t)-cut.*

Using this lemma, it is easy to prove the correctness of Algorithm 7.1. Let R be a supernode of $\mathcal{T}$ and let $R\,S_1, \ldots, R\,S_l$ be the edges of $\mathcal{T}$ which are incident to R. For $i = 1, \ldots, l$, let $(U_i, V(G) \setminus U_i)$ denote the cut in G induced by the cut-tree edge $R\,S_i$, where U_i is chosen to contain S_i. Further, let G_R denote the graph which results from G by identifying the node sets U_i, $i = 1, \ldots, l$, to single nodes each, which we denote by s_i, $i = 1, \ldots, l$.

7.1.4 Theorem ([GH61]) *Let $\mathcal{T}$ be a cut-tree with terminal node set $T \neq V(G)$ and let $t \in V(G) \setminus T$, let R be the supernode which contains t and let $r \in T$ be its representative. If $(X, \bar{X})$ is a minimum (r,t)-cut in G_R, then construct a new tree $\mathcal{T}'$ from $\mathcal{T}$ in the following way. Construct the tree $\mathcal{T}'$ from $\mathcal{T}$ by replacing the supernode R by the two supernodes $R_1 := R \cap X$ and $R_2 := R \setminus X$ of $\mathcal{T}$ linked by an edge with weight*

$$f_{R_1 R_2} := \lambda(G_R, r, t) = \lambda(G, r, t).$$

For $i = 1, \ldots, l$, add an edge $R_1\,S_i$, if $s_i \in X$, or $R_2\,S_i$, if $s_i \in \bar{X}$, to $\mathcal{T}$. The weight of this edge is f_{RS_i}. Then the resulting tree $\mathcal{T}'$ is a cut-tree with terminal node set $T' := T \cup \{t\}$.

Below we will need the following easy lemma.

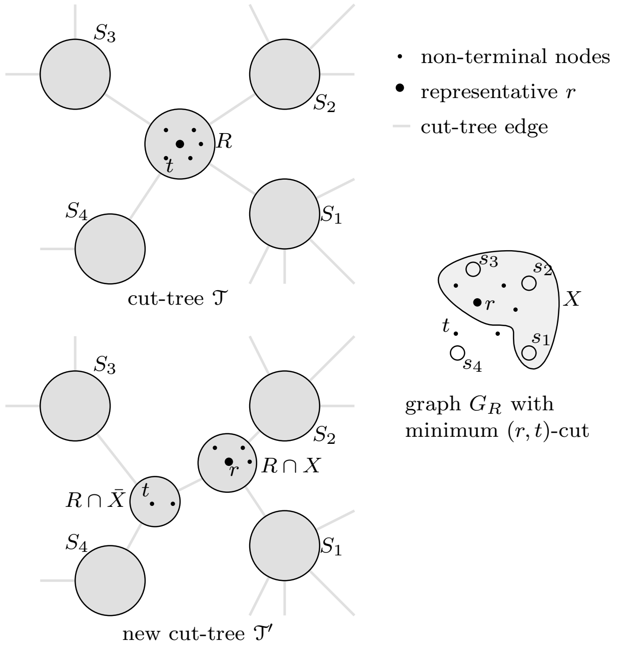


Figure 7.1: Core step of the Gomory-Hu cut-tree algorithm

7.1.5 Lemma *Let $\mathcal{T}$ be a cut-tree for G, AB an edge of $\mathcal{T}$ and s and t the representatives of A and B respectively. Let $(S, V(G) \setminus S)$ be the minimum (s,t)-cut induced by AB (with $s \in S$). The component of $\mathcal{T} \setminus AB$ which contains A (B) is a cut-tree for $G/(V(G) \setminus S)$ (G/S) with terminal node set $T \cap S$ ($T \setminus S$).* □

7.1.2 Equivalent flow trees and minimum (s,t)-cuts

Now we will briefly describe an algorithm due to Gusfield & Naor [GN93], which produces a data structure which contains, for any two nodes $s, t \in T$, *all* minimum (s,t)-cuts, and it only takes the time required to compute $|T| - 1$ max-flows. The data structure has $O(|T|n)$ space requirement.

The question of finding and storing all minimum (s,t)-cuts for a fixed pair s,t was treated by Picard & Queyranne [PQ80]: if all strongly connected components in the residual network of a maximum (s,t)-flow are shrunk, we obtain a directed acyclic graph. We call the components *SCCs*, and we call the shrunk directed acyclic graph *DAG*. We note the following well-known fact. *SCC,DAG*

7.1.6 Lemma ([PQ80]) *Let $\mathrm{DAG}_{(s,t)}$ be the directed acyclic graph of a maximum (s,t)-flow. Then an (s,t)-cut $(S, V(G) \setminus S)$ is minimum if and only if each strongly connected component of the residual network is contained in either S or $V(G) \setminus S$ and there are no arcs in D crossing from S to $V(G) \setminus S$.* □

Consequently, we call $\mathrm{DAG}_{(s,t)}$ the *DAG-representation* of all minimum (s,t)-cuts in G. Note that in some definitions of DAG-representation the direction of the arcs is reversed.

For finding the minimum (s,t)-cuts for *all* pairs s,t, the central result is the following lemma, which relates it to equivalent flow trees. A consequence of the lemma is that Algorithm 7.2 solves that problem of finding all minimum (s,t)-cuts for all pairs s,t.

7.1.7 Lemma ([GN93]) *Let $(S, V(G) \setminus S)$ be a minimum (s,t)-cut, and let $\mathcal{T}$ be an arbitrary equivalent flow tree for G. There exists an edge (u,v) on the path from s to t in $\mathcal{T}$ with the property that $(S, V(G) \setminus S)$ is a minimum (u,v)-cut.*

Algorithm 7.2 Gusfield-Naor

```
Input: Undirected connected graph $G$ with positive edge weights and a set of terminal nodes
    $T \subseteq V(G)$.
Output: Equivalent flow tree $\mathcal{T}$, and for every edge $\{r, s\}$ of $\mathcal{T}$ the DAG representation of all
    minimum $(r, s)$-cuts, $\mathrm{DAG}_{(r,s)}$.
 1: Find an order of the nodes in $T$: $t_0, \ldots, t_{|T|-1}$.
 2: Let $\mathcal{T}$ be a star with $t_0$ at the center and leaves $t_1, \ldots, t_{|T|-1}$.
 3: For $k = 1, \ldots, |T| - 1$ do
 4:   Let $r$ be the (unique) neighbor $r$ of $t_k$ in $\mathcal{T}$.
 5:   Compute $\mathrm{DAG}_{(t_k,r)}$, associate it to the edge $\{t_k, r\}$ of $\mathcal{T}$.
 6:   For $j = k + 1, \ldots, |T| - 1$ do
 7:     If the (unique) neighbor of $t_j$ is $r$, and $t_j$ is in the $t_k$-set of $\mathrm{DAG}_{(t_k,r)}$ then
 8:       Disconnect $t_j$ from $r$, and connect it to $t_k$.
 9:     End if
10:   End for
11: End for
12: Output $\mathcal{T}$ and the DAGs associated to its edges.
```

7.1.3 Representability of minimum odd cuts and minimum odd (s, t)-cuts

We say that a cut $(S, V(G) \setminus S)$ is *representable,* if there exists a pair of nodes s, t such that $(S, V(G) \setminus S)$ is a minimum (s, t)-cut. We note the following facts for easy reference.

7.1.8 Lemma *Let $(U, V(G) \setminus U)$ be a minimum odd cut. The following statements are equivalent:*

(i). $(U, V(G) \setminus U)$ is representable.

(ii). There exists a node $s \in T \cap U$ and a node $t \in T \setminus U$ such that there exists an odd minimum (s, t)-cut.

(iii). There exists a node $s \in T \cap U$ and a node $t \in T \setminus U$ such that $\lambda(s, t) = c(U)$ holds. □

The following theorem is an immediate and well-known consequence of the correctness of the Padberg-Rao minimum odd cut algorithm [PR82].

7.1.9 Theorem *Every minimum odd cut is representable.* □

A similar statement holds for minimum odd (s, t)-cuts again. By Lemma 7.1.7 and the following Lemma 7.1.10, Algorithm 7.2 can be used to compute a minimum T-odd (s, t)-cut in time $O(|T|nm \log(n^2/m))$.[1] The result will also be needed in the context of minimum (s, t)-blossoms.

7.1.10 Lemma *[GR95] Let $x, y \in V(G)$ and $(U, V(G) \setminus U)$ be a T-odd (T-even) cut separating x and y with minimal capacity among these cuts. Then there exist $s \in U$ and $t \in V(G) \setminus U$ such that $(U, V(G) \setminus U)$ is a minimum (s, t)-cut.*

7.2 The Minimum T-odd cut problem

The following basic lemma is a variant of the uncrossing lemma in [PR82]. We recall that two cuts $(X, V(G) \setminus X)$ and $(Y, V(G) \setminus Y)$ are said to *cross,* if none of the four sets $X \cap Y$, $Y \setminus X$, $X \setminus Y$, $V(G) \setminus (X \cup Y)$ is empty. *crossing cuts*

[1] But see 10.3.3 in the chapter on the adaption of algorithmic techniques to the Branch-and-Cut environment.

7.2.1 Lemma *If $(U, V(G) \setminus U)$ is a minimum odd cut and $(S, V(G) \setminus S)$ a minimum (s,t)-cut, for arbitrary s, t in $V(G)$, and U crosses S, then one of the following four sets is a minimum odd cut: $S \cap U$, $S \setminus U$, $U \setminus S$, $V(G) \setminus (S \cup U)$.*

Proof. By renaming the sets, we can assume that $S \cap U$ is odd. If $t \notin U$, we can conclude by submodularity that $S \cap U$ is a minimum odd cut:

$$c(S \cap U) \leq c(S) + c(U) - c(S \cup U) \leq c(U).$$

If $t \in U$ and $S \setminus U$ is odd, then the same argument holds for $S \setminus U$. Thus assume that $t \in U$ and $V(G) \setminus (S \cup U)$ is odd. If $s \in U$, then the argument works for $V(G) \setminus (S \cup U)$. Otherwise, $(S \cup U, V(G) \setminus (S \cup U))$ is an odd cut and $(U, V(G) \setminus U)$ is an (s,t)-cut, hence

$$c(S \cap U) \leq c(S) + c(U) - c(S \cup U) \leq c(S) \leq c(U).$$

□

As a consequence of this lemma we obtain the key idea of the Padberg-Rao method for computing a minimum odd cut.

7.2.2 Theorem ([PR82]) *Let $\mathcal{T}$ be any cut-tree with a terminal node set T' containing all odd nodes. Then there exists an edge AB of $\mathcal{T}$ whose removal induces a minimum odd cut in G.*

Proof. The proof is by induction on $|T|$. If $|T| = 2$, then the statement is obviously correct. For $|T| \geq 4$, we chose any edge AB of $\mathcal{T}$, let s, t be the representatives of its end nodes, and denote by $(S, V(G) \setminus S)$ the minimum (s,t)-cut induced by AB. By Lemma 7.2.1, there exists a minimum odd cut not crossing $(S, V(G) \setminus S)$. Further, if $(S, V(G) \setminus S)$ is odd, then either the edge AB induces a minimum odd cut, or no minimum odd cut separates s and t. In the latter case, or if the cut $(S, V(G) \setminus S)$ is even, we conclude by induction, invoking Lemmas 7.1.5, where, if $(S, V(G) \setminus S)$ is odd, the node s ($\bar{s}$) in G/S ($G/(V(G) \setminus S)$) is even. □

7.2.1 A new algorithm for minimum odd cuts

The starting point of our new algorithm is the following easy fact.

7.2.3 Lemma *Let s,t be two nodes such that there exists an odd minimum (s,t)-cut. Then*

$$\lambda^1(G) = \min\left(\lambda(G,s,t), \lambda^1(G \circ \{s,t\})\right).$$

□

Given the DAG-representation of all minimum (s,t)-cuts, it is easy to decide if there exists an odd minimum (s,t)-cut. The following easy lemma gives the correspondence.

7.2.4 Lemma *There exists an odd minimum (s,t)-cut if and only if there is an odd SCC in the DAG-representation of the minimum (s,t)-cuts.* □

By applying Lemma 7.2.1 inductively, we can refine it to the SCCs in the DAG-representation.

7.2.5 Lemma *Let $(U, V(G) \setminus U)$ be a minimum odd cut, and let $S_0, \ldots, S_k$ be the SCCs in the DAG-representation of all the minimum (s,t)-cuts in reverse topological order. Suppose that none of the SCCs is odd. There exists an $l \in \{1, \ldots, k\}$ for which $S_l \cap U$ and $S_l \setminus U$ are odd sets. For each such l, there exists a minimum odd cut $(U', V(G) \setminus U')$ with the following two properties:*

1. *$U' \cap S_l = U \cap S_l$ and $S_l \setminus U' = S_l \setminus U$, and*
2. *for all $j \neq l$ either $S_j \subseteq U'$ or $S_j \subseteq V(G) \setminus U'$.* □

Algorithm 7.3 Recursive min odd cut

Input: Graph G, capacities c, and set of odd nodes T, $|T| \geq 2$ and even.
Output: A minimum odd cut.

```
1: Initialize the value of the best cut found so far $\phi := \infty$.
2: Loop
3:   Let $s$ and $t$ be two distinct odd nodes.
4:   Compute a minimum $(s,t)$-cut.
5:   Create the DAG-representation of all minimum $(s,t)$-cuts. Sort it reverse topologically
     and search it for an odd minimum $(s,t)$-cut.
6:   If an odd minimum $(s,t)$-cut is found then
7:     If it is the best cut found so far, store it and update $\phi$.
8:     Shrink $s$ and $t$ into a single node. If no odd nodes remain, return the best odd cut found
       and Stop.
9:   Else
10:    Leave the loop.
11:  End if
12: End loop
13: Let $S_0, \ldots, S_k$ denote the sets of the DAG in topological ordering.
14: For $l = 0, \ldots, k$ do
15:   If $S_l$ contains at least two odd nodes then
16:     Construct a graph $G'$ by shrinking in $G$ each of the two node sets $\bigcup_{i=0}^{l-1} S_i$ and $\bigcup_{i=l+1}^{k} S_i$
        to single nodes.
17:     By recursion, compute a minimum odd cut in $G'$.
18:     If a minimum odd cut with value $< \phi$ was found then
19:       Store the odd cut and update $\phi$.
20:     End if
21:   End if
22: End for
23: Return the best odd cut.
```

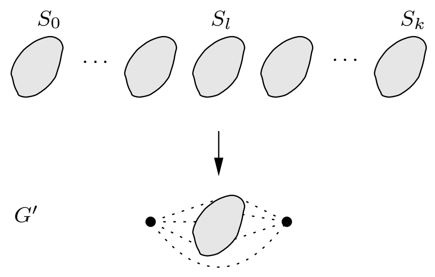


Figure 7.2: Recursive minimum odd cut computation for SCCs

Algorithm 7.4 Simple min odd cut with shrinking

Input: Graph G, capacities c, and set of odd nodes T, $|T| \geq 2$ and even, upper bound ϕ.
Output: The value $\lambda^1{}_c(G,T)$, if it is strictly less than ϕ.

```
1:  While odd nodes left do
2:    Shrink edges (see 7.2.2) as long as possible. If all odd nodes have vanished, leave the loop.
3:    Choose odd nodes $s$ and $t$ and compute a minimum $(s,t)$-cut $(S, V(G) \setminus S)$.
4:    If $(S, V(G) \setminus S)$ is an odd cut or $\lambda(s,t) \geq \phi$ then
5:      Update $\phi := \lambda(s,t)$.
6:      Identify $s$ and $t$.
7:    Else
8:      Recursively apply the algorithm to $G/S$ and $G/(V(G) \setminus S)$, update $\phi$ if necessary.
9:      Leave the loop.
10:   End if
11: End while
12: If the value of $\phi$ was improved by the algorithm, return it.
```

These three lemmas together prove the correctness of Algorithm 7.3. It works by selecting a pair of odd nodes s,t, constructing the DAG-representation of all minimum (s,t)-cuts, and checking if one of the SCCs is odd. If that is the case, s and t are identified to a new even node. Otherwise the algorithm is recursively applied to each SCC S_l, with the SCCs S_i, $i \leq l-1$, and S_i, $i \geq l+1$ identified to two single nodes. See Fig. 7.2. A variant of this algorithm is to compute only one minimum (s,t)-cut instead of the DAG-representation. In Algorithm 7.4, we give a full-fledged algorithm which also uses shrinking techniques which we now describe.

7.2.2 Shrinking and minimum odd cuts

The shrinking rules we propose can be implemented to be very fast if the graph has few edges. The following shrinking rules are adopted from [PR90b].

7.2.6 Lemma *Let uv be an edge of G. Under any of the following conditions, the edge uv can be contracted without increasing the value of the minimum odd cut.*

(a). u is even, and $c(\partial(u)) \leq 2c_{uv}$, or

(b). both u and v are even, and there is a node $w \in N(u) \cap N(v)$ such that

$$c(\partial(u)) \leq 2c_{uv} + 2c_{uw}, \quad \text{and} \quad c(\partial(v)) \leq 2c_{uv} + 2c_{vw}.$$

Proof. If an odd cut contains the edge uv, then an odd cut with smaller capacity can be constructed by moving u or v to the other side. □

These simple rules could be generalized, but they turn out to be effective in practice. The first one can be checked in time $O(n+m)$, the second requires $\sum_u (\deg u)^2$ steps. We now come to shrinking rules which are based on the value of the best odd cut found so far, which is recorded by the Algorithm 7.4 in the variable ϕ. The following lemma gives the simple idea.

7.2.7 Lemma *Let u,v be two nodes and $q_{u,v}$ be a lower bound on the capacity of a minimum (u,v)-cut in G. If $q_{u,v} \geq \phi$, then u,v can be identified without increasing the capacity of the minimum odd cut.*

As an example, define $\gamma := 0$, if u and v are not connected by an edge and $\gamma := c_{uv}$, otherwise. Then $\gamma + \sum_{w \in N(u) \cap N(v)} \min(c_{uw}, c_{wv})$ is a lower bound on $\lambda(u,v)$. For another way to apply Lemma 7.2.7, we build on results of [NOI94].

7.2.8 Definition A *legal ordering* of the node set of a graph G is an ordering $v_1, \dots, v_n$ with the property that the edge $v_k v_{k+1}$ is a minimum capacity edge in the cut $\partial(\{v_1, \dots, v_k\})$.

A legal ordering can be constructed by an algorithm in [NOI94], called MCAP, in time $O(n \log n)$. It has the following property, which makes it useful in the context of minimum (odd) cuts.

7.2.9 Lemma *Let $v_1, \dots, v_n$ be a legal ordering. Then $q_k := c(\{v_1, \dots, v_k\}) \le \lambda(v_k, v_{k+1})$, and $q_{n-1} = \lambda(v_{n-1}, v_n)$.*

In our minimum odd cut algorithm, we invoke MCAP as a shrinking mechanism. MCAP works by iteratively selecting for v_{k+1} the node which minimizes c_e, $e \in \partial(\{v_1, \dots, v_k\})$. The algorithm also computes the numbers q_k, which are checked for the condition of Lemma 7.2.7. The version of MCAP which is most useful for minimum odd cuts will prefer even nodes in the selection of v_{k+1} if the minimum is attained in more than one value, in the hope that the node v_n is odd, which would allow v_{n-1} and v_n to be identified (Lemma 7.2.3). If during the iteration one of the sets $\{v_1, \dots, v_k\}$ is odd and $c(\{v_1, \dots, v_k\})$ happens to be smaller than ϕ, a new best odd cut is found. A drawback of the use of MCAP is that it cannot be assured that one of the successful cases occurs, i.e., the time spent on an individual call to MCAP might be in vain. In Chapter 11, we give computational results which show that this does not happen too often.

Given that both the core minimum odd cut algorithm and the shrinking operations are fairly simple, it is astounding how far an only moderately fine-tuned implementation outperforms the fastest cut-tree algorithms known to us. We refer to Section 11.1.

7.3 Blossom minimization

In this section we define what we call the "blossom minimization" problem, and variants. Given a graph G and an even cardinality set $T \subseteq V(G)$, a *blossom* is a pair (U, F) consisting of a non-empty node set $U \subsetneq V(G)$, together with a set of *flipped* edges $F \subseteq \partial(U)$ with the property that the number $|T \cap U| + |F|$ is odd. Given two weight functions $E(G) \to \mathbb{Q}_+$, the *normal* weight c and the *flipped* weight c', the value of a blossom (U, F) is defined to be $c(\partial(U) \setminus F) + c'(F)$. We can now consider the problems of finding a minimum (i.e., minimum value) blossom, or a blossom which is minimum under the additional constraint that it must separate two given nodes, i.e., a minimum "(s,t)-blossom". In this section we consider these two problems, and we also touch on the subject of finding all minimum blossoms. Blossom minimization is a natural extension of the so-called (capacitated) blossom separation problem [PR82], which historically requires characteristic conditions, namely that c and c' satisfy $c_e + c'_e \ge 1$ for all $e \in E(G)$, and that only blossoms of value strictly less than 1 are of interest. The possibility of dropping these conditions turns out to be useful in practice: in 8.2.1 we can abandon the requirement that, prior to invoking the separation routine for (switched) simple 2-regular PB-inequalities, we must make sure that no violated connectivity inequality exist. We start with a survey of known results on blossom separation.

blossom
flipped edges
normal/flipped weight

7.3.1 Known results on blossom separation

We recall the definition for the *(capacitated) blossom separation problem* from [PR82]. The input is a multigraph G, an even cardinality set $T \subseteq V(G)$, and two vectors $x\colon E(G) \to \mathbb{Q}_+$ and $u\colon E(G) \to \mathbb{Z}_+^*$, where it can be assumed that $x \le u$ holds. The objective is to find a cut $\partial(U)$ with $U \subseteq V(G)$ and a set of flipped edges $F \subseteq \partial(U)$ such that $|T \cap U| + |u|(F)$ is an odd number, and $x(\partial(U) \setminus F) + (u - x)(F) < 1$ holds—or to prove that no such cut exists. In their seminal paper, Padberg & Rao [PR82] give a polynomial time combinatorial algorithm for blossom separation. We say that x_e is the normal value of the edge e, and $u_e - x_e$ is the flipped value.

The Padberg-Rao algorithm produces the so-called *split graph* $\hat{G}$ by subdividing each edge e of G by a *splitting node*. Of the two edges, one gets weight x_e and even parity, while the other gets weight $u_e - x_e$ and odd parity. The algorithm then proceeds by computing a minimum $\hat{T}$-odd cut in $\hat{G}$, where

$$\hat{T} := T \triangle \triangle_{\substack{e\in E(\hat{G})\\ \text{odd}}} \epsilon(e)$$

(here, $\epsilon(e)$ is the two-element set consisting of the end-nodes of e). The running time amounts to $|\hat{T}| - 1$ max-flow computations on $\hat{G}$, i.e., $O(|\hat{T}||V(\hat{G})||E(\hat{G})| \log(|V(\hat{G})|^2/|E(\hat{G})|)) = O(|\hat{T}|m^2 \log n)$. The cardinality of $\hat{T}$ depends on u and T, but can be $\Theta(m)$ (if, for example, $u = \mathbf{1}$), and hence the worst-case running time of the Padberg-Rao algorithm is $O(m^3 \log n)$.

Grötschel & Holland [GH87] found an easy way to speed up the Padberg-Rao algorithm (however, according to the remarks in [GH87], it is quite difficult to implement): when computing a minimum odd cut, before computing each max-flow, the subdivision of the edges can be undone, or, more generally, one of two edges incident to a node with degree two can be contracted (the one with bigger weight). It was noted by Letchford (see [LRT04]) that this modification reduces the worst case running time to $|\hat{T}| - 1$ max-flows on G (rather than $\hat{G}$), which results in $O(nm^2 \log(n^2/m))$.

Padberg & Rinaldi [PR90a] proposed a heuristic algorithm for blossom separation which runs in $O(n^2 m \log(n^2/m))$ time. In a personal communication to us, Letchford conjectured the correctness of an algorithm which is a very slight modification (which is actually even easier) of the heuristic in [PR90a], and which has the same worst case running time. In [LRT04] we were able to prove the correctness of this algorithm based on the construction of Padberg & Rao [PR82], while here we give a proof which is based on uncrossing, and which shows that Letchford's algorithm can be used for blossom minimization. Our proof is analogous to the natural proof of correctness of the min T-odd cut algorithm in Section 7.2.

7.3.2 Blossom minimization

Algorithm 7.5 Blossom minimization

Input:
Multigraph G, set $T \subseteq V(G)$, and weights $c, c' \colon E(G) \to \mathbb{Q}_+$.

Output:
A set U minimizing (7.1)

```
1:  Initialize the variable $m_{\min} := \infty$. It holds the value $\beta(U)$ of the best set $U$ found so far.
2:  Set $w := \min(c, c')$.
3:  Compute a cut-tree for the graph $G$ with set of terminal nodes $V(G)$ by using any cut-tree
    algorithm.
4:  **For** each of the $n - 1$ edges of the cut-tree **do**
5:    Let $\partial(U)$ denote the cut induced by the cut-tree edge.
6:    Compute $\beta(U)$.
7:    **If** $\beta(U) < m_{\min}$ **then**
8:      Store $U$, set $m_{\min} := m_U$.
9:    **End if**
10: **End for**
11: Output $U$.
```

7.3.1 Definition Let G be a graph, $T \subseteq V(G)$ with $|T|$ even, and c, c' weight functions $E(G) \to \mathbb{Q}_+$. We define the function

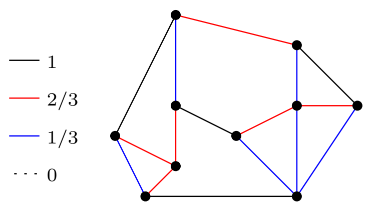
β

$$\beta \colon \mathbf{2}^{V(G)} \setminus \{\emptyset, V(G)\} \to \mathbb{Q}_+ :$$
$$U \mapsto \beta(U) := \min\{c(\partial(U) \setminus F) + c'(F) \mid F \subseteq \partial(U) \wedge |T \cap U| + |F| \text{ odd}\} \quad (7.1)$$

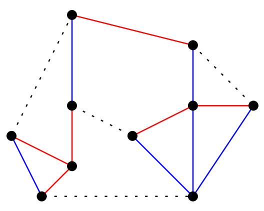
The *blossom minimization problem* is the following. Given G, T and c, c' as above, find a set U which minimizes β.

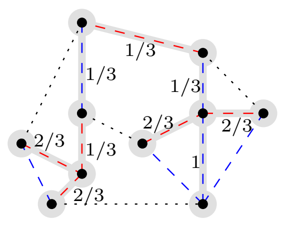
We note that β is not in general submodular. Clearly, the blossom separation problem can be solved via blossom minimization (for the issue of parallel edges see Lemma 7.3.2 below).

Letchford conjectured that Algorithm 7.5 solves the blossom minimization problem. The algorithm is illustrated on an example in Fig. 7.3. The input graph is displayed in 7.3(a), 7.3(b) shows G with the weights constructed in step 2. The cut-tree is shown in 7.3(c). Two example iterations of the loop in steps 4–10 are illustrated in 7.3(d)–7.3(f), and 7.3(g)–7.3(h).

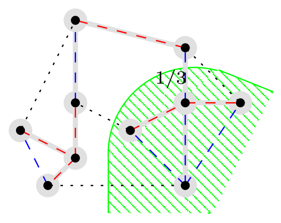


(a) *Graph G with weights c; $c' := \mathbf{1} - c$.*

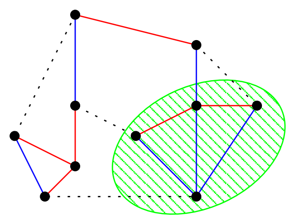

(b) *Graph G with weights* $\min(c, c')$.

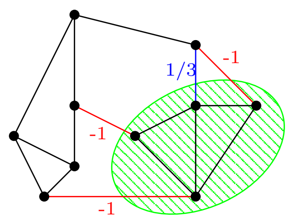


(c) *Cut-tree for G with weights* $\min(c, c')$.

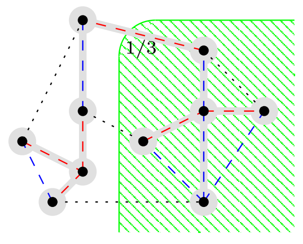


(d) *An edge of the cut-tree induces a cut W.*

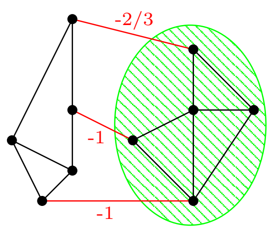

(e) *The cut W in the graph G.*



(f) *The blossom (W, F) where F (bold) is the arg-min in* (7.1).



(g) *Another cut W' induced by a cut-tree edge.*



(h) *Another blossom (W', F') with minimum F'.*

Figure 7.3: Illustration for Algorithm 7.5

Because of Lemma 7.3.3 for step 6, its running time is dominated by the computation of a cut-tree for G with terminal node set $V(G)$, which can be done by $n-1$ max-flow computations [GH61]. Hence, the running time of Algorithm 7.5 is $O(n^2 m \log({}^{n^2}\!/_m))$. With the focus on blossom separation, this is not only an improvement over the previously best known algorithm by Grötschel et al. [GH87], which has running time $O(nm^2 \log({}^{n^2}\!/_m))$, but Algorithm 7.5 is also very simple and easy to implement (compared to the admittedly difficult to implement algorithm in [GH87]).

In what follows, we will prove that Algorithm 7.5 does in fact solve the blossom minimization problem.

7.3.3 Preparatory facts

The following simple results are implicit in [PR90a].

7.3.2 Lemma *Given a set of edges E, the two values*

$$\min\Big\{\sum\nolimits_{e\in E\setminus F} c_e + \sum\nolimits_{f\in F} c'_f \;\Big|\; F\subseteq E,\ |F| \text{ even/odd}\Big\} \tag{7.2}$$

and their argument sets F can be computed in time $O(|E|)$.

Proof. One of the two sets is $F := \{e\in E \mid c'_e < c_e\}$, and the other is $F\triangle\{f\}$, where f minimizes $|c'_e - c_e|$ over all $e\in E$. $\square$

7.3.3 Lemma *Given U, $\beta(U)$ can be computed in time $O(|\partial(U)|)$.* $\square$

7.3.4 Lemma *We may assume w.l.o.g. that $c\le c'$ holds.*

Proof. If necessary, for $e = uv$, exchange c_e and c'_e while replacing T by $T\triangle\{u,v\}$. $\square$

7.3.5 Lemma *Suppose that $c\le c'$. If $U\subsetneq V(G)$ with $U\ne\emptyset$ is a T-even set, then there exists an edge $f\in\partial(U)$ such that $\beta(U) = c(\partial(U)\setminus\{f\}) + c'_f$. Or, in other words, for all $U\subsetneq V(G)$ with $U\ne\emptyset$ (even or odd) we have*

$$\beta(U) = \begin{cases} \min_{f\in\partial(U)} c(\partial(U)\setminus\{f\}) + c'_f & \text{if } U \text{ is } T\text{-even,}\\ c(U) & \text{if } U \text{ is } T\text{-odd.}\end{cases} \tag{7.3}$$

$\square$

7.3.4 All minimum blossoms

Now we give some arguments which lead to an $O(n^2 m\log(n^2/m))$-time algorithm to identify *all* minimizers of β. However, it is much more complicated than Algorithm 7.5. We first prove the following fact. For convenience, we use the abbreviation $c^{(f)} := c + (c'_f - c_f)\chi^f$ for an edge $f\in E(G)$. $c^{(f)}$

7.3.6 Lemma *Let U minimize β. There exists $s\in U$ and $t\in V(G)\setminus U$ such that U is a w-minimum (s,t)-cut.*

Proof. We may assume that $c\le c'$. If $\min_U \beta(U) = \lambda^1{}_c(G)$, then the statement follows from Theorem 7.1.9. If $\min_U\beta(U) < \lambda^1{}_c(G)$, then there is an $f\in\partial(U)$ such that $(U, V(G)\setminus U)$ is a minimum odd cut in G with respect to the weights $c^{(f)}$, and hence, there exist $s\in U$, $t\in V(G)\setminus U$ such that $(U, V(G)\setminus U)$ is a $c^{(f)}$-minimum (s,t)-cut with $f\in\partial(U)$. This implies

$$\lambda_c(s,t) \ge \lambda_{c^{(f)}}(s,t) - c'_f + c_f = c(U),$$

and hence $(U, V(G)\setminus U)$ is a c-minimum (s,t)-cut. $\square$

Note that we cannot assume that $s,t\in T$. Given a DAG representation $\mathrm{DAG}_{(s,t)}$ of all w-minimum (s,t)-cuts, the sets U minimizing β subject to $(U,V(G)\setminus U)$ separating s and t can be found in the following way. First all parities of SCCs of $\mathrm{DAG}_{(s,t)}$ must be computed, and if one is odd, then every odd cut stored in $\mathrm{DAG}_{(s,t)}$ minimizes β over all minimum (s,t)-cuts. Otherwise, iteratively take an edge $f\in E(G)$ which crosses between different SCCs and compute the value $\lambda_c(s,t) - c_f + c'_f$. If it is small enough, then every cut stored in $\mathrm{DAG}_{(s,t)}$ which is odd if the parities of the SCCs containing the end nodes of f are flipped minimizes β over all minimum (s,t)-cuts.

Clearly all minimizers can be found in this way. We note that, from the arguments of [GN93] (see Lemma 7.1.7), it follows that, to identify all sets U which minimize β without the restriction that a particular pair of nodes must be separated, it is sufficient to check $n-1$ DAGs.

In Algorithm 7.5, we do not have all minimum (s,t)-cuts per pair, we just have one. To prove the correctness of Algorithm 7.5, we need an uncrossing argument which is stronger than Lemma 7.3.6.

7.3.5 An uncrossing result

7.3.7 Lemma *Let U minimize β, and for two nodes s, t of G let $(S,\bar S)$ be a c-minimum (s,t)-cut in G. With $\bar U := V(G)\setminus U$, (at least) one of the following five sets minimizes β:*

$$S, \quad \begin{array}{ll} S\cap U, & S\cap \bar U, \\ \bar S\cap U, & \bar S\cap \bar U. \end{array}$$

Proof. Clearly, we can assume that S and U are crossing. Suppose first that U is odd. Then U is a minimum odd cut, and the statement of the lemma follows from Lemma 7.2.1.

Now let U be even. We assume w.l.o.g. that (7.3) holds, i.e., $c \le c'$. Let f be an edge which minimizes the term $c(\partial(U)\setminus\{f\}) + c'_f$ in (7.3). We have to distinguish cases.

1. *$s,t \in U$:* Since $S\cap U$ is even iff $\bar S\cap U$ is, and the same holds for U replaced by $\bar U$, this case is entirely symmetric w.r.t. exchanging S and $\bar S$ (and s and t). We have the following sub-cases.

   (a) *$S\cap U$, $\bar S\cap U$, $S\cap\bar U$, $\bar S\cap\bar U$ even:*

      i. *$f\in E(S)$:* Since $S\cap\bar U$ is even and $c(S)\le c(S\cup\bar U)$, we have by submodularity of $c^{(f)}(\,\cdot\,)$
      $$\begin{aligned}\beta(S\cap\bar U) \le c^{(f)}(S\cap\bar U) &\le c^{(f)}(S) + c^{(f)}(\bar U) - c^{(f)}(S\cup\bar U) \\ &= c(S) + c^{(f)}(\bar U) - c(S\cup\bar U) \le c^{(f)}(\bar U).\end{aligned} \tag{7.4}$$

      ii. *$f\in(S\cap U : \bar S\cap\bar U)$:* Let $f=uv$, where $u\in U$. We claim that $(S:\bar S)$ is a $c^{(f)}$-minimal $(\{s,u\},\{t,v\})$-cut. If this is proven, then, by submodularity, and because $\partial((\bar S\cup\bar U))$ is a $(\{s,u\},\{t,v\})$-cut, it follows that $\bar S\cap\bar U$ is a set minimizing β:
      $$\beta(\bar S\cap\bar U) \le c^{(f)}(\bar S\cap\bar U) \le c^{(f)}(\bar S) + c^{(f)}(\bar U) - c^{(f)}(\bar S\cup\bar U) \le c^{(f)}(\bar U). \tag{7.5}$$
      We now prove the claim. Let λ denote the value of a $c^{(f)}$-minimal $(\{s,u\},\{t,v\})$-cut. Clearly $\lambda\le c^{(f)}(S)$. For every $(\{s,u\},\{t,v\})$-cut $(R,\bar R)$ we have
      $$c(S)\le c(R) = c^{(f)}(R) - c'_f + c_f.$$
      which implies $c(S)\le \min_R(c(R)-c'_f+c_f) = \lambda - c'_f + c_f$, and hence $c^{(f)}(S)\le\lambda$, which proves the claim.

   (b) *$S\cap U$, $\bar S\cap U$ even, $S\cap\bar U$, $\bar S\cap\bar U$ odd:* We have
   $$\beta(\bar S\cap\bar U) \le c(\bar S\cap\bar U) \le c(\bar S) + c(\bar U) - c(\bar S\cup\bar U) \le c(\bar U) \le c^{(f)}(\bar U). \tag{7.6}$$
   This implies that the odd set $\bar S\cap\bar U$ minimizes β.

   (c) *$S\cap U$, $\bar S\cap U$ odd, $S\cap\bar U$, $\bar S\cap\bar U$ even:*

      i. *$f\in E(S)$:* Analogous to the case 1(a)i.

      ii. *$f\in(S\cap U : \bar S\cap\bar U)$:* Analogous to the case 1(a)ii.

   (d) *$S\cap U$, $\bar S\cap U$, $S\cap\bar U$, $\bar S\cap\bar U$ odd:* Analogous to the case 1b.

2. *$s\in U$, $t\in\bar U$:*

(a) *$S \cap U$, $\bar{S} \cap U$, $S \cap \bar{U}$, $\bar{S} \cap \bar{U}$ even:*

  i. *$f \in E(S)$:* The argument is the same as in 1(a)i but with $S \cap \bar{U}$ replaced by $S \cap U$.

  ii. *$f \in E(T)$:* The argument is the same as in 1(a)i but with $S \cap \bar{U}$ replaced by $\bar{S} \cap \bar{U}$.

  iii. *$f \in (S \cap U : \bar{S} \cap \bar{U})$:* Analogous to the case 1(a)ii.

  iv. *$f \in (S \cap \bar{U} : \bar{S} \cap U)$:* Since $(U : \bar{U})$ is an (s,t)-cut, we have $c(S) \leq c(U)$. Since $f \in \partial(S) \cap \partial(U)$, it follows that

$$c^{(f)}(S) = c(S) - c_f + c'_f \leq c(U) - c_f + c'_f = c^{(f)}(U). \tag{7.7}$$

  Since S is even, we have $\beta(S) \leq c^{(f)}(S) \leq c^{(f)}(U)$, whence S is a set minimizing β.

(b) *$S \cap U$, $\bar{S} \cap U$ even, $S \cap \bar{U}$, $\bar{S} \cap \bar{U}$ odd:* Since $\partial(U)$ is an (s,t)-cut, we have

$$c(S) \leq c(U) \leq c^{(f)}(U) = \beta(U),$$

whence S is an odd set minimizing β.

(c) *$S \cap U$, $\bar{S} \cap U$ odd, $S \cap \bar{U}$, $\bar{S} \cap \bar{U}$ even:* The same as in the previous case 2b.

(d) *$S \cap U$, $\bar{S} \cap U$, $S \cap \bar{U}$, $\bar{S} \cap \bar{U}$ odd:* Analogous to 1b.

□

7.3.8 Theorem *Algorithm 7.5 is correct, i.e., it minimizes β over $\mathbf{2}^{V(G)} \setminus \{\emptyset, V(G)\}$.*

Proof. First note again that we can w.l.o.g. assume that $c \leq c'$, i.e, $w = c$. The proof is by induction on $n := |V(G)|$. The case that $n = 2$ is trivial.

To conclude $n \rightsquigarrow n+1$, let $\{A, B\}$ be any edge of the cut-tree, where $A =: \{s\}$ $B =: \{t\}$, and $(S, \bar{S})$ the corresponding c-minimum (s,t)-cut. If S minimizes β, we are done.

Otherwise, by Lemma 7.3.7, there exists a set U minimizing β, which is contained, w.l.o.g., in S, whence $|S| \geq 2$. We construct the loopless multigraph $G_S := G \circ (\bar{S} \cup \{s\})$. We denote the node replacing $\bar{S} \cup \{s\}$ by $\hat{s}$, and the restricted weights by c_S and c'_S. It is easy to see that the cut-tree resulting by replacing the part of the cut-tree on the B-side of the edge $\{A, B\}$ by the single supernode $\{\hat{s}\}$ is a cut-tree for G_S with set of representatives $S \cup \{\hat{s}\} \setminus \{s\}$ and weights $\min(c_S, c'_S)$. By induction, steps 4–10 of Algorithm 7.5 find a set minimizing β on this graph. But for each set $U \subseteq V(G_S)$ with $U \not\ni \hat{s}$ the value of β when computed with respect to c_S and c'_S in G_S coincides with the value $\beta(U)$ when computed with respect to c, c' in G. Hence the induction is complete, and the correctness of Algorithm 7.5 follows. □

7.3.6 Consequences

We note the proof of correctness via Lemma 7.3.7 is completely analogous to the natural proof of the correctness of the Padberg-Rao minimum odd cut algorithm (the proof in [PR82] is a bit less direct). Thus, our result not only puts capacitated blossom separation on the same theoretical running time level as uncapacitated blossom separation (which coincides with minimum odd cut), but it also shows that capacitated blossom separation (or blossom minimization) is in fact “the same” as uncapacitated blossom separation (or minimum odd cut), except that the parities of the nodes or edges are not known *a priori*. Hence it is not surprising that Algorithm 7.5 is in direct analogy with the Padberg-Rao minimum odd cut algorithm.

Another parallel is that there is a “blossom variant” of the minimum odd cut Algorithm 7.3. Namely, the uncrossing result implies the correctness of Algorithm 7.6 for blossom minimization, which is not based on a cut-tree. In this recursive algorithm, nodes are marked, and it is first invoked with all nodes unmarked. Theoretically, shrinking mechanisms as in Algorithm 7.4 are possible, though they appear to be less promising.

Algorithm 7.6 Recursive blossom minimization

Input:
Multigraph G, set $T \subseteq V(G)$, weights c, c' with $c \leq c'$, and set of marked nodes.

Output:
A set U minimizing (7.1)

```
1:  While there exist at least two unmarked nodes do
2:    Let $s, t$ be two unmarked nodes of $G$.
3:    Compute a minimum $(s,t)$-cut $(S, \bar{S})$ in $G$.
4:    If $(S, \bar{S})$ is an odd cut then
5:      If $\lambda(s,t)$ is small enough, store the blossom $(S, \emptyset)$.
6:      Identify $s, t$ to a new unmarked node.
7:    Else
8:      Compute $\beta(S)$.
9:      Construct the graph $G_{\bar{S}}$ by identifying the node set $S$ to a new node $v_s$. Parallel edges
        are merged (see Section 7.1). The node $v_s$ is marked, and it inherits the parity of the
        set $S$. Recursively compute a minimum blossom in $G_{\bar{S}}$.
10:     Do the same with the set $\bar{S}$.
11:     Store the one of the three blossoms of steps 9–11 with the smallest value of $\beta$.
12:     Leave the while-loop.
13:   End if
14: End while
15: Return the best blossom stored.
```

7.3.7 Minimum (s,t)-blossoms

Now we propose the problem of finding a minimum blossom separating two given nodes and show that the problem can be solved in time $O(n^2 m \log(n^2/m))$. The algorithm will be applied in Section 8.3 for the separation of certain subclasses of (switched) path bridge inequalities. Our construction is based on the following simple fact.

7.3.9 Proposition *Let $w := \min(c, c')$. For $x, y \in V(G)$, there exists $s \in U$ and $t \in V(G) \setminus U$ and a w-minimum (s,t)-cut $(U, V(G) \setminus U)$ such that U minimize β among all cuts separating x and y. Given a DAG-representation of all minimum (s,t)-cuts, such a U can be found in time $O(nm)$. Further, only the $n-1$ DAGs computed by Algorithm 7.2 need to be considered.*

Proof. We may assume that $c \leq c'$. Let U be any set separating x and y and minimizing β subject to this condition. If $\min_U \beta(U) = \lambda^1{}_c(G)$, then the statement follows from Lemma 7.1.10. If $\min_U \beta(U) < \lambda^1{}_c(G)$, then $(U, V(G) \setminus U)$ is a minimum odd cut in G with respect to the weights $c^{(f)}$, and hence, there exist $s \in U$, $t \in V(G) \setminus U$ such that $(U, V(G) \setminus U)$ is a $c^{(f)}$-minimum (s,t)-cut with $f \in \partial(U)$. This implies

$$\lambda_c(s,t) \geq \lambda_{c^{(f)}}(s,t) - c'_f + c_f = c(U),$$

and hence $(U, V(G) \setminus U)$ is a c-minimum (s,t)-cut.

To identify a set U in a DAG-representation of all minimum (s,t)-cuts, we first compute the parities of all SCCs in $\mathrm{DAG}_{(s,t)}$. Any odd cut separating both s and t and x and y minimizes β. After $O(n+m)$ preprocessing, we can find such a cut, if one exists, in time $O(n)$. If none exists, we iterate over all edges $f \in E(G)$ which cross between different SCCs of $\mathrm{DAG}_{(s,t)}$, flip the parities of the SCCs containing the end nodes of f, and check for an odd cut separating both s and t and x and y, which can again be done in $O(n)$. Among the odd (x,y)-cuts found in this way, we take one for which the value $\lambda_c(s,t) - c_f + c'_f$ is minimum.

The fact that only $n-1$ DAGs have to be considered follows from Lemma 7.1.7. □

7.3.10 Corollary *There exists an $O(n^2 m \log(n^2/m))$ algorithm to compute a minimum (s,t)-blossom.* □

Chapter 8

Exact separation algorithms

In this chapter we mainly apply the results of the previous chapter to some of the separation problems occurring in cutting-plane approaches to the General Routing Problem. In the first section we briefly review some known results on separation for the GRP. The following sections contain our contributions. In Section 8.2, we deal with switched path bridges and in particular improve a polynomial time separation algorithm for the simple 2-PB inequalities of Letchford [Let96] in three ways: we extend it to a bigger class of inequalities, namely the switched simple 2-PB, we improve the running time by a factor of ${}^{m}\!/_{n}$, and we obtain the practical advantage that the algorithm can be used without the precondition that the point x^* must satisfy all connectivity inequalities (when the precondition is violated, we are not guaranteed to find a violated inequality if one exists). In Section 8.3 we give polynomial time separation algorithms for subsets of (switched) path bridge inequalities in the presence of special conditions on x^*, in particular we present an algorithm for KC-separation relying on the cactus representation of all minimum cuts of a graph.

In this chapter, $\Gamma = (G, \mathcal{C}, \mathbf{t})$ will denote a GRP-structure, $b \colon E(G) \to \mathbb{Z}_+ \cup \{\infty\}$ a vector of upper bounds, and $x^* \colon E(G) \to \mathbb{Q}$ a rational vector. We will always assume that x^* satisfies all non-negativity and upper bound inequalities $\mathbf{0} \leq x^* \leq b$. We denote by $G(x^*)$ the spanning subgraph of G with edge set $\{e \in E(G) \mid x^*_e \neq 0\}$. We always implicitly use the restriction of x^* as edge weights on $G(x^*)$. By $G^{\mathrm{s}}_{\mathcal{C}}(x^*)$ (and $G^{\mathrm{m}}_{\mathcal{C}}(x^*)$ resp.) we denote the simple graph (or loopless multigraph, resp.) which results from $G(x^*)$ by identifying the R-sets to nodes while merging (or keeping, resp.) parallel edges; its edge weights are defined by x^* in the way described in 0.2.2. Further, unless otherwise stated, we will use the abbreviations $n := |V(G)|$, $m := |E(G(x^*))|$, $\dot{n} := |V(G_{\mathcal{C}})|$ and $\dot{m} := |E(G^{\mathrm{s}}_{\mathcal{C}}(x^*))|$.

8.1 Known results

It is well-known that connectivity inequalities (1.1b) can be separated exactly by computing a minimum cut in the graph $G^{\mathrm{s}}_{\mathcal{C}}(x^*)$, and it is also completely apparent that the separation problem for odd-cut inequalities can be solved by computing a minimum odd cut in $G(x^*)$ [CS94]. This is typically performed by the uncapacitated blossom separation algorithm of Padberg & Rao [PR82] (or avoided due to the computational effort of the procedure, cf. [GL00]), but clearly our Algorithm 7.4 described in Section 7.2 can be used. There is no need to discuss the issue at this time; instead we refer to 10.3.1, where we relate some details of the implementation of Algorithm 7.4 and to Section 11.1, where we report on computational results.

A polynomial time exact separation routine for simple 2-regular PB-inequalities is introduced by Letchford in [Let96]. It is a variant of the capacitated blossom separation algorithm of Padberg & Rao [PR82]. Using the Grötschel & Holland [GH87] modification of the Padberg-Rao algorithm, Letchford's separation routine can be implemented to run in time $O(nm^2 \log({}^{n^2}\!/_{m}))$.

The Padberg-Rao algorithm requires that $x^0 + x^1 \geq \mathbf{1}$. If that is not the case, the algorithm may produce results which cannot be interpreted as violated inequalities, in our case, simple 2-

regular PB-inequalities. In the way [Let96] applies the Padberg-Rao algorithm, the condition $x^0 + x^1 \geq \mathbf{1}$ is satisfied if the connectivity inequality $x(\partial(U)) \geq 2$ is satisfied for all sets $U = \{u\}$, $U = \{v\}$ and $U = \{u, v\}$, for any pair of adjacent R-isolated nodes u, v. If shrinking operations are performed prior to invoking the separation procedure, the conditions can become more restrictive. Whenever they fail to hold, the separation of simple 2-regular path bridge inequalities must be skipped, which can become a notable drawback.

8.2 Switched PBs

In this section we first show how any separation algorithm for classical path-bridge inequalities can be used to separate the class of switched path-bridge inequalities with $F \subseteq E_{\text{int}}(G)$ (Section 3.1). Then we give a new exact polynomial time separation algorithm for the subclass of switched simple PBs. We start with a lemma.

8.2.1 Lemma *If x^* satisfies all connectivity inequalities* (1.1b) *and violates a classical PB-inequality, then $x^*(A : Z) < 1$ holds, where A and Z refer to the sets of the PB-configuration as defined in 3.3.2.* □

Proof. For $p = 1, \dots, P$, let E_p be the set of edges of G which cross between different sets B_i^p, B_j^p, $1 \leq i < j \leq n$ and define $T_p := \bigcup_{j=1}^{n} B_j^p$. We have

$$2x(E_p) + 2x(\partial(T_p)) = \sum_{j=1}^{n} x^*(\partial(B_j^p)) + x^*(\partial(T_p)) \geq 2(n_p + 1).$$

Following [Nad02], for $l = 1, \dots, \prod_p (n_p - 1)$, let

$$H_l := \bigcup_{\substack{p=1,\dots,P,\\ j=1,\dots,n_p,\\ (j-1)(n_p-1) \leq l}} B_j^p.$$

The path bridge inequality can be written as $\sum_l x(\delta(H_l)) + \sum_p \prod_{q \neq p} (n_q - 1)\, x(\delta(T_p)) \geq \prod_p (n_p - 1) + \sum_p (n_p + 1) \prod_{q \neq p} (n_q - 1)$. We have

$$\begin{aligned}
\sum_l x^*(\partial(H_l)) &+ \sum_p \prod_{q \neq p} (n_q - 1)\, x^*(\partial(T_p)) \\
&\geq \sum_l x^*(\partial(H_l) \setminus E_p) + \sum_p \prod_{q \neq p} (n_q - 1) \left(x^*(\partial(T_p)) + x^*(E_p) \right) \\
&\geq \sum_l x^*(\partial(H_l) \setminus E_p) + \sum_p (n_p + 1) \prod_{q \neq p} (n_q - 1).
\end{aligned}$$

Since x^* violates the PB-inequality, and noting that $(A : Z) \subseteq \partial(H_l) \setminus E_p$ holds for all p, j, we summarize

$$\begin{aligned}
\prod_p (n_p - 1) + \sum_p (n_p + 1) \prod_{q \neq p} (n_q - 1) &> \sum_l x^*(\partial(H_l)) + \sum_p \prod_{q \neq p} (n_q - 1)\, x^*(\partial(T_p)) \\
&\geq \prod_p (n_p - 1)\, x^*(A : Z) + \sum_p (n_p + 1) \prod_{q \neq p} (n_q - 1),
\end{aligned}$$

whence $x^*(A : Z) < 1$. □

From the GRP-structure $\Gamma = (G, \mathcal{C}, \mathbf{t})$ and a vector x^* satisfying all connectivity inequalities, we can derive a GRP-structure $\hat{\Gamma} = (\hat{G}, \hat{\mathcal{C}}, \hat{\mathbf{t}})$ and a vector $\hat{x}^*$, which satisfies all connectivity inequalities on $\hat{\Gamma}$, with $\left|V(\hat{G})\right| \leq |V(G)| + |E_{\text{int}}(\Gamma)|$ and $\left|E(\hat{G})\right| \leq |E(G)| + |E_{\text{int}}(\Gamma)|$ which has the property that x^* violates a switched PB-inequality with $F \subseteq E_{\text{int}}(\Gamma)$ if and only if $\hat{x}^*$ violates a classical PB-inequality defined on $\hat{\Gamma}$. Thus we can use any separation algorithm (even heuristic) on $\hat{\Gamma}$ and $\hat{x}^*$ to produce a violated switched PB-inequality.

For every $e \in E_{\text{int}}(\Gamma)$, we denote by $\psi^0(e), \psi^1(e) \in V(G)$ the two end nodes of e. We construct the graph $\hat{G}$ from G by *splitting* every edge $e \in E_{\text{int}}(\Gamma)$ by replacing it with a new node k_e and two edges $\{\psi^0(e), k_e\}$, $\{k_e, \psi^1(e)\}$. We let $\hat{x}^*_{\{\psi^0(e),k_e\}} = x^*_e$ and $\hat{x}^*_{\{k_e,\psi^1(e)\}} = b_e - x^*_e$. For $e \notin E_{\text{int}}(\Gamma)$, we define $\hat{x}^*_e = x^*_e$. The R-sets $\hat{\mathcal{C}}$ are constructed from $\mathcal{C}$ by inserting every split node k_e in the same set which contains the end nodes of e. Fig. 8.1 gives an illustration.

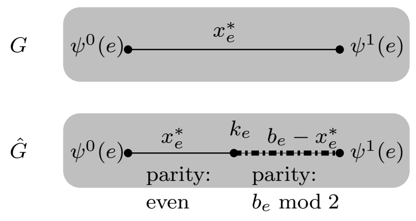


Figure 8.1: Construction of $\hat{G}$ from G

Next we assign parities to the edges of $\hat{G}$. We call the edge $\{k_e, \psi^1(e)\}$ *odd* for all $e \in E_{\text{int}}(\Gamma)$ for which b_e is odd. All other edges are called *even.* A node receives odd parity iff it is incident to an odd number of odd edges. In other words, if for $v \in V(G)$ we let r_v denote the number of odd edges incident to v in $\hat{G}$, we define the new parities by

$$\hat{\mathbf{t}}(v) := \begin{cases} \mathbf{t}(v) + r_v \mod 2, & \text{if } v \in V(G), \\ b_e \mod 2, & \text{if } v = k_e \text{ for an } e. \end{cases}$$

We have defined a GRP-structure $\hat{\Gamma}$, and $\hat{x}^*$ satisfies all connectivity constraints.

8.2.2 Proposition *A switched path-bridge inequality with $F \subseteq E_{\text{int}}(\Gamma)$ on Γ is violated by x^* if and only if $\hat{x}^*$ violates a classical path-bridge inequality on $\hat{\Gamma}$.*

Proof. Let $\hat{A}$, $\hat{Z}$, $\hat{B}^p_j$ be a PB-configuration on $\hat{\Gamma}$ and $\hat{a}z \geq \hat{\alpha}$ be the corresponding PB-inequality which is violated by $\hat{x}^*$. We let $A := \hat{A} \cap V(G)$, $Z := \hat{Z} \cap V(G)$, and $B^p_j := \hat{B}^p_j \cap V(G)$. Further we denote by F the set of edges of G such that the edge $\{k_e, \psi^1(e)\}$ of $\hat{G}$ is in $(\hat{A} : \hat{Z})$. By virtue of the previous lemma, a switched PB-inequality is defined in this way, and the slack of the inequality equals $\hat{\alpha} - \hat{a}\hat{x}^*$.

On the other hand, if x^* violates a switched PB-inequality, a PB-inequality on $\hat{G}$ violated by $\hat{x}^*$ can be constructed in the same straightforward manner. $\square$

Using the algorithm mentioned in Section 8.1, the class of the switched simple 2-regular PB-inequalities with $F \subseteq E_{\text{int}}(\Gamma)$ can be separated in polynomial time if there exists no violated connectivity inequality. Letchford's algorithm yields a running time of $O(\hat{n}\hat{m}^2 \log(\hat{n}^2/\hat{m}))$, where $\hat{n}$ is the number of nodes and $\hat{m}$ is the number of edges of $\hat{G}$. In the next section, we will show how the class of *all* switched simple 2-regular PB-inequalities, i.e., without restriction on the set F, can be separated in even better worst case time, and without the precondition that all connectivity inequalities must be satisfied.

8.2.1 Switched simple 2-PBs

We now propose a separation algorithm for switched simple 2-regular PB-inequalities, see Fig. 8.2. It runs in time $O(n^2 m \log(n^2/m))$. Let a handle $H \subseteq V(G)$, and teeth $T_1, \ldots, T_P \subseteq V(G)$, $p \geq 0$, and a set of edges $F \subseteq \partial(H)$ be given. Assume that the following conditions hold

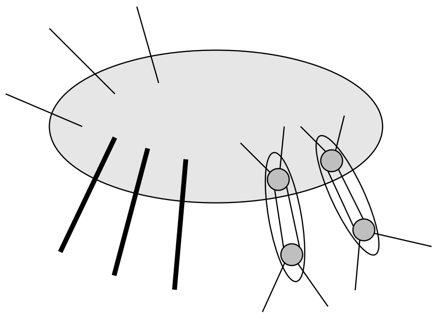


Figure 8.2: Switched simple 2-regular path bridge inequality (bold edges are switched).

1. $\mathbf{t}(H) + P + |F| = 1 \mod 2$.
2. $T_p = \{u, v\}$ for R-isolated nodes $u \in H$ and $v \in V(G) \setminus H$, and $E(T_p) \neq \emptyset$.
3. The $P + 1$ sets F, $E(T_p)$, $p = 1, \ldots, P$ are pairwise disjoint.

The inequality

$$x(\partial(H) \setminus F) - x(F) + \sum_{p=1}^{P} x(\partial(T_p)) \geq 3P + 1 - b(F) \tag{8.1}$$

is valid for GRP(Γ, b). The inequalities of this type include the switched 2-regular PB-inequalities. By a technical but standard uncrossing argument, it is possible to obtain a violated switched 2-regular PB-inequality or connectivity inequality from a violated inequality (8.1).

For the separation of (8.1), we define a simple graph G^s by removing from G all but one edge in each set of parallel edges of G. This means that G^s has node set $V(G^s) = V(G)$ and $uv \in E(G^s)$ if and only if u and v are neighbors in G. Suppose that $u, v \in V(G)$ are adjacent. We define

$$\theta_{uv} := \begin{cases} \max\bigl(0,\ x^*(\partial(\{u,v\})) + x^*(u:v) - 3\bigr), & \text{if } u \text{ and } v \text{ are both R-isolated, and} \\ \infty & \text{otherwise.} \end{cases}$$

As pointed out in [Let96], $x^*(\partial(\{u,v\})) + x^*(u:v) - 3 \geq 0$ holds if all connectivity inequalities are satisfied. Now we let

$$\begin{aligned} c^0_{uv} &:= \min_{\substack{F \subseteq (u:v) \\ b(F) \text{ even}}} \bigl(x^*\bigl((u:v) \setminus F\bigr) + b(F) - x^*(F)\bigr), \\ c^1_{uv} &:= \min\Bigl(\theta_{uv},\ \min_{\substack{F \subseteq (u:v) \\ b(F) \text{ odd}}} \bigl(x^*((u:v) \setminus F) + b(F) - x^*(F)\bigr)\Bigr). \end{aligned} \tag{8.2}$$

As an immediate consequence of these definitions, we have the following proposition.

8.2.3 Proposition *Every blossom (W, F) with $c^0(\partial(W) \setminus F) + c^1(F) < 1$ gives a violated switched simple 2-regular PB inequality. If for every two adjacent R-isolated nodes u, v each of the tree values $x^*(\partial(u)), x^*(\partial(v)), x^*(\partial(\{u,v\}))$ is at least two, then the reverse direction also holds, i.e., if there exists a violated switched simple 2-regular PB inequality, then there exists such a blossom.* □

We note that $c^0_{uv} + c^1_{uv} \geq 1$ for all $uv \in E(G)$. We could use the blossom separation algorithm for capacitated matching problems of Padberg & Rao [PR82], which requires computing $O(|E(G^s)|)$ maximum flows on a graph with $O(|E(G)|)$ edges in the worst case. Using the algorithm described in Section 7.3, this problem can be solved in the time which is required to perform n max-flow computations on G^s.

8.2.4 Corollary *The switched simple 2-regular path-bridge inequalities can be separated in time $O(n^2 m \log(n^2/m))$.* □

In practice, we would not expect to find many pairs u, v with more than one edge in $b(u : v) \geq 2$. This may be different, if, prior to invoking the separation routine, shrinking operations have been performed on the graph, for example those described in [CLS01] for the classical simple 2-regular PB-inequalities. Another possibility is to separate *switched simple 2-regular path inequalities,* which are the special case of switched simple 2-regular PB-inequalities when both A and Z are unions of R-sets (compare [CFN85, CS98]). By using the above algorithm after shrinking each R-set into a single node, we obtain the following result.

8.2.5 Corollary *The class of switched simple 2-regular path inequalities can be separated in time $O(\dot{n}^2 \dot{m} \log(\dot{n}^2/\dot{m}))$.* □

8.3 Some results about KC- and PB-separation

In this section we will apply the results of the previous chapter to the separation of path bridge and KC-inequalities. Our results will also prove useful for the heuristic separation of path bridge inequalities in the following chapter.

8.3.1 Definition We say that a tuple $P = (B_1, \dots, B_n)$, with $n \geq 2$, consisting of disjoint unions of R-sets is an *oriented path* of length n. When we talk of *non-oriented* paths, we identify the tuples $(B_1, \dots, B_n)$ and $(B_n, \dots, B_1)$. *(non-)oriented path*

A path is called a *simple 2-path,* if its length is two, and the two sets B_1, B_2 consist of one R-set each, which in turn consists of one node each, in other words, if $|B_1| + |B_2| = 2$ holds. We also speak of a *simple tooth.* *simple 2-path/tooth*

8.3.1 Separation of 2-PBs with a bounded number of "big" teeth

As a first application we transfer a result of Carr [Car97] for the separation of comb inequalities with a fixed number of teeth for STSP to the GRP. We also enlarge the class of inequalities which are separated, and we improve the running time. We start with a definition which improves the one in [Car97].

8.3.2 Definition Let $H, T_1, \dots, T_k, \dots, T_P$ be the handle and teeth of a (switched) 2-regular PB-inequality, where $T_{k+1}, \dots, T_P$ are simple teeth (possibly $k = P$). A *backbone* of the inequality with respect to x^* is a set of k pairs of *adjacent* R-sets, which can be ordered $(C_1, D_1), \dots, (C_k, D_k)$, such that $C_j \subseteq T_j \cap H$ and $D_j \subseteq T_j \setminus H$ for each $j = 1, \dots, k$.

The crucial observation for the improvement of the running time is the following.

8.3.3 Remark If a violated (switched) 2-regular PB inequality with handle H which has a tooth T such that $(T \cap H : T \setminus H) = \emptyset$, then there must exist violated connectivity or non-negativity (or upper-bound) inequalities.

As a consequence, if x^* satisfies all connectivity and bound inequalities, then every (switched) 2-regular PB-inequality violated by x^* has a backbone with respect to x^*. Further, there exist less than $2^{k-1}\binom{\dot{m}}{2}$ backbones which belong to violated (switched) 2-regular PB-inequalities with k non-simple teeth (note exchanging C_j with D_j for all j gives a backbone of the same 2-PB inequality).

Now let x^* satisfy all connectivity and bound inequalities, and let a violated (switched) 2-regular PB-inequality be given. Following [Car97] we let $(C_1, D_1), \dots, (C_k, D_k)$ be any of its backbones with respect to x^*, in the ordering mentioned in the definition. For each j, let $\partial(S_j)$ be a cut with $C_j, D_j \subseteq S_j$ and $C_i, D_i \subseteq V(G) \setminus S_j$ for $i = 1, \dots, k$, $i \neq j$, and for which the value $x^*(\partial(S_j))$ is minimum with respect to these conditions. These sets can be obtained by performing k minimum (s, t)-cut computations if $k \geq 2$, or one minimum cut computation if $k = 1$. Note that

$x^*(\partial(S_j)) \le x^*(\partial(T_j))$. As is shown in [Car97], it can be assumed w.l.o.g., that the S_j are pairwise disjoint, since the value $\sum_j x^*(\partial(S_j))$ does not increase by the obvious uncrossing operation.

At this point we again take a different road than [Car97]. Namely, we let (W', F) denote a minimum (s,t)-blossom in the graph G' obtained from G by shrinking the R-sets contained in the sets S_j to single nodes each, and then identifying the nodes obtained by shrinking the R-sets $C_1, \dots, C_k$ to a single new node s, the nodes obtained by shrinking the R-sets $D_1, \dots, D_k$ to a new node t. The nodes s and t are declared odd, all other nodes keep their parities from Γ. The weights c^0 and c^1 for the edges which are not adjacent to s or t are defined as in (8.2), the c^0 for the edges adjacent to s or t are the x^*-values while their c^1 values are set to ∞. We denote by H' the node set of G' defined by H, and we note again that $c^0(W \setminus F) + c^1(F) \le \beta(H')$. Hence, when we define $W := W' \cup \{s_1, \dots, s_k\} \setminus \{s\}$, we see that the handle W and the teeth $S_1, \dots, S_k$ together with the teeth defined by edges in the set F define a violated switched 2-regular PB-inequality. We have proven the following result.

8.3.4 Proposition *Let x^* satisfy all connectivity (and bound) inequalities. The class of all (switched) 2-regular path bridge inequalities with at most $k \ge 2$ non-simple teeth can be separated in the time which is required to perform $k\binom{m}{k}$ max-flow (minimum (s,t)-cut) computations on $G^{\mathrm{s}}_{\mathcal{C}}(x^*)$ and $2^{k-1}\binom{m}{k}$ minimum (s,t)-blossom computations on $G(x^*)$.* $\square$

For the special case of (switched) KC-inequalities, we have the following consequence.

8.3.5 Corollary *(Switched) KC-inequalities with $K = 3$ can be separated in time $O(n^2 m^2 \log(n^2/m))$.* $\square$

8.3.2 Separation of PBs with a fixed set of paths

Let $\vec{\mathcal{P}}$ be a set of pairwise disjoint oriented paths, i.e., no two sets B_i of distinct paths intersect. It is easy to see that, by computing a minimum odd (s,t)-cut, we can test if there exists a violated PB-inequality using precisely these paths in their orientations, and, if possible, produce a set A which completes the PB-configuration. By computing a minimum (s,t)-blossom instead of a minimum (s,t)-cut, we can test for the existence of a violated switched PB-inequality which uses all these paths, and possibly some simple 2-paths. Both of these algorithms work by deleting from G all nodes belonging to one of the paths, and adding two new nodes s and t. The weights of the edges in this graph can be chosen such that the odd (s,t)-cuts (or (s,t)-blossoms) with value strictly less than one correspond to violated PB-inequalities.

What is more interesting is that, if we impose restrictions on the LP-solution x^*, we can drop both the condition that the orientation of the paths is fixed, and the condition that the PB-configuration of a violated inequality must contain all paths. Let $\mathcal{P}$ be a set of pairwise disjoint non-oriented paths. Denote by $\mathscr{C}(\mathcal{P})$ the class of path bridge inequalities on configurations $\mathcal{C}$ with the following property:

(a). The set of paths of $\mathcal{C}$ is a subset of $\mathcal{P}$.

(b). If $P \in \mathcal{P}$ is not a path of $\mathcal{C}$ then P is entirely contained in either the set A or the set Z of $\mathcal{C}$.

Further, let $\mathscr{C}_*(\mathcal{P})$ denote the set of path bridge inequalities on configurations $\mathcal{C}$, for which (b) holds, and (a) holds only for the paths of $\mathcal{C}$ which are not simple 2-paths. The classes which result from $\mathscr{C}(\mathcal{P})$ and $\mathscr{C}_*(\mathcal{P})$ by allowing switching of edges in $(A : Z)$ will be referred to as the switched classes, and denoted by $\mathscr{C}'(\mathcal{P})$ and $\mathscr{C}'_*(\mathcal{P})$. We note that the class $\mathscr{C}(\mathcal{P})$ contains R-odd cut inequalities and $\mathscr{C}_*(\mathcal{P})$ contains simple 2-PB-inequalities.

8.3.6 Proposition *Let a set of non-oriented paths $\mathcal{P} = \{(B^p_1, \dots B^p_{n_p}), p = 1, \dots, P\}$ be given. Suppose that x^* satisfies all connectivity inequalities. If*

$$x^*(B^p_i : B^q_j) = 0 \qquad \text{for all } p,q,i,j \text{ with } \quad p \ne q \wedge \big(2 \le i \le n_p - 1 \quad \vee \quad 2 \le j \le n_q - 1\big) \tag{8.3}$$

then the subclass $\mathscr{C}(\mathcal{P})$ of path bridge inequalities can be separated by an algorithm with worst-case running-time $O(n^2 m \log(n^2/m))$, and the same holds for the classes $\mathscr{C}_*(\mathcal{P})$, $\mathscr{C}'(\mathcal{P})$ and $\mathscr{C}'_*(\mathcal{P})$.

Proof. Algorithm 8.1 shows how the class $\mathscr{C}(\mathcal{P})$ is separated. It can easily be blended with the constructions in 8.2.1 to separate over the other classes. Since x^* satisfies all connectivity inequalities, the edge weights computed in step 2 are non-negative. The weights are chosen in such a way that the minimum odd cuts correspond to violated inequalities in the class $\mathscr{C}(\mathcal{P})$, if the value of a minimum odd cut is strictly less than one.

Note that in contrast to [CS94], our definition of KC-inequality allows a "degenerate" KC-configuration in which the set A (or, equivalently, Z) is empty. However, such an inequality cannot be defined by a *minimum* odd cut, if there is no violated connectivity inequality. □

We now turn to the special case of KC-inequalities.

8.3.3 Circular KCs

A variant of Algorithm 8.1 is Algorithm 8.2, which checks a single path for a violated (switched) KC-inequality. This algorithm too can be used to check for a violated *switched* KC in the same worst-case running time, if the minimum odd (s,t)-cut computation in step 6 is replaced by a minimum (s,t)-blossom computation; see Proposition 7.3.9. Since the condition (8.3) is vacuous for KCs, the only question is how candidate paths can be found. We now deal with the special case of what we call circular (switched) KCs.

8.3.7 Definition Let $x^* \in \mathbb{Q}_+^{E(G)}$. We say that a (switched) KC-inequality on the beads $B_0, \dots, B_K$ is *circular* (with respect to x^*), if $A_0 := B_0 \cup B_K, A_1 := B_1, \dots, A_{K-1} := B_{K-1}$ forms a circular partition of $G^{\mathrm{s}}_{\mathcal{C}}(x^*)$, (see 0.4.1). *circular KC*

The circular KCs include what one might call "maximally violated" KCs, which are circular KCs with the additional properties that $x^*(B_0 : B_1) = x^*(B_{K-1} : B_K) = 1$ and $x^*(B_0 : B_K) = 0$. If there does not exist a violated connectivity inequality, i.e., if $\lambda_{x^*}(G^{\mathrm{s}}_{\mathcal{C}}(x^*)) = 2$, we can use the cactus representation of all minimum cuts of $G^{\mathrm{s}}_{\mathcal{C}}(x^*)$ for separation of circular (switched) KC-inequalities. Algorithm 8.3 displays the algorithm for circular KC-inequalities. We will use the idea of this algorithm for the design of a separation heuristic for KC-inequalities in the next chapter.

Algorithm 8.1 Check list of non-oriented paths for a violated (switched) PB-inequality

Input:

GRP-structure $\mathit{\Gamma}$, LP-solution x^* satisfying all connectivity inequalities, and a list of paths $(B_1^p, \dots, B_{n_p}^p)$, $p = 1, \dots, P$.

Output:

May produce a violated PB-inequality using a subset of the paths.

```
1: Let $V_0$ be the set of nodes of $G$ which are not contained in a path. Construct the subgraph
   of $G$ induced by $V_0$, and add to this graph $2P$ new nodes $s_1, t_1, \dots, s_P, t_P$. Denote the
   resulting graph by $H$. The weight of an edge of $H$ is the $x^*$-value of the edge.
2: Add edges $s_p t_p$ to $H$ with weights
   $$-\frac{n_p+1}{n_p-1} + \frac{1}{n_p-1} x^*(\partial(\bigcup_j B_j^p)) + \sum_{1 \le i < j \le n_p} \frac{j-i}{n_p-1} x^*(B_i : B_j),$$
   for $p = 1, \dots, P$.
3: For all $p$ and $v \in V_0$ for which $(\bigcup_j B_j^p : v)$ is not empty do
4:    Add edges $s_p v$ and $t_p v$ to $H$, with weights
      $$\sum_{j=1}^{n_p} \frac{n_p - j}{n_p - 1} x^*(B_j^p : v) \quad \text{and} \quad \sum_{j=1}^{n_p} \frac{j-1}{n_p-1} x^*(B_j^p : v) \quad \text{resp.}$$
5: End for
6: For all edges $e \in E(G_{\mathcal{C}}^{\mathrm{m}})$ which cross between different paths do
7:    Suppose that $B_j^p$ and $B_1^q$ are the beads containing the end nodes of $e$. (The case $B_{n_q}^q$ is
      symmetric.)
8:    If there are not yet edges $s_p s_q$ and $s_p t_q$ in $H$, add the missing edges, and initialize their
      weights to zero.
9:    To the weight of the edge $s_p s_q$ add the number
      $$\sum_j \frac{n_p - j}{n_p - 1} x^*(B_j^p : B_1^q).$$
      For the edge $s_p t_q$ this is symmetric.
10: End for
11: Assign to each node $v \in V_0$ the parity $\mathbf{t}(v)$. The nodes $s_p$ and $t_p$ are all odd.
12: Compute a minimum odd cut in $H$. If $\lambda^1(H) < 1$, then every minimum odd cut corresponds
    to a violated inequality in the class $\mathscr{C}(\mathcal{P})$.
```

Algorithm 8.2 Check a path for a violated (switched) KC-inequality

Input:

GRP-structure $\varGamma$, LP-solution $x^* \in \mathbb{Q}_+^{E(G)}$, and a path $(B_1, \ldots, B_{K-1})$, i.e. an ordered list of pairwise disjoint sets.

Output:

A violated KC-inequality (possibly degenerate, if violated connectivity inequalities exist) using the path, iff one exists.

```
1: Denote $V_0 := V(G) \setminus \bigcup B_i$.
2: Construct the subgraph $H$ of $G(x^*)$ induced by $V_0$.
3: Add two new nodes $s$ and $t$ to $H$. Add an edge $st$ with weight $-1 + \sum_{1 \le i < j \le K-1} x^*(B_i : B_j)$.
4: For every edge $uv \in (B_i : V_0)$ with $u \in B_i$ and $v \in V_0$, add an edge $sv$, with weight $\frac{K-i}{K-2} x_e^*$,
   and an edge $tv$ with weight $\frac{i}{K-2} x_e^*$.
5: Let $T := \{s, t\} \cup \{v \in V_0 \mid \mathbf{t}(v) = 1\}$.
6: Compute a minimum $T$-odd $(s, t)$-cut in $H$
7: There exists a violated KC-inequality if and only if $\lambda^1(G, s, t) < 1$, and the sets $B_0$ and $B_K$
   can be constructed from a minimum $T$-odd $(s, t)$-cut in the obvious way.
```

Algorithm 8.3 Circular KC-separation

Input:

GRP-structure $\varGamma$, LP-solution $x^* \in \mathbb{Q}_+^{E(G)}$ satisfying all connectivity inequalities and cactus representation of all minimum cuts of $G_{\mathbb{C}}^{\mathrm{s}}(x^*)$.

Output:

A violated circular (switched) KC-inequality, if one exists.

```
1: For all circles of the cactus do
2:   Let $A_0, \ldots, A_{l-1}$ be the beads of the circular partition.
3:   For $k = 1, \ldots, l-2$ do
4:     For $i = 0, \ldots, l-1$ do
5:       Check the path $(A_{i+k+1}, \ldots, A_{i-1})$, where indices are taken modulo $l$.
6:       If a violated KC-inequality is found, Stop.
7:     End for
8:   End for
9: End for
10: There exists no violated circular KC-inequality.
```

Chapter 9

Separation heuristics

In our search for separation heuristics, we considered the classes of inequalities which have proven to be useful by both theoretical analyses and practical experience: path bride and honeycomb classes and their intersection, the KCs. For the theoretical value of path bridge inequalities, we refer to the result of Goemans [Goe95], which was transferred to the GRP by Letchford [Let04]. The practical value of KCs and honeycombs has been demonstrated in [CLS01]. It is also known that the addition of path bridge inequalities can improve the lower bounds in a cutting plane approach considerably [Let04].

Separation heuristics for KCs in which $B_0 \cup B_K$ consists of one or two R-sets, for regular PB-inequalities, and for honeycombs with $L = 1$ and where $B_0 \cup \ldots \cup B_{n_1}$ consists of one or two R-sets are described and tested in [BCS00, CLS01]. According to a personal communication from A. Letchford and A. Corberán, the KC- and HC-algorithms work extremely well, while the PB-algorithm leaves much to be desired. All these heuristics first find R-sets (or pairs of R-sets) which are split into two or more parts, and then try to construct the paths which complete the path bridge or honeycomb inequalities.

In this chapter we propose a number of new separation algorithms for KC-, HC-, and (not necessarily regular) PB-inequalities. Our aim was to pursue new ideas rather than to work on existing approaches. We pursue an opposite strategy to that of [BCS00, CLS01]: we first search for paths and then find matching sets A (for path bridges and KCs) or $B_1^1, \ldots, B_n^1$ (for honeycombs). The algorithms we propose follow one or both of the following basic intuitive guidelines.

1. Split the separation problem into two subproblems, where one can be solved quickly and exactly by a polynomial time algorithm. One of the two parts produces "raw material", while the other part tries to put together the pieces.

2. Instead of looking at the LP-solution only, let the algorithm be guided by the structure which is exposed in a cactus representation of all minimum cuts.

9.1 KC-heuristic based on circular partitions

This heuristic for KC-separation is based on Algorithm 8.3 for circular KCs in the previous chapter. With the results of this section we contribute to the search for practicable algorithms for KC-separation in which the set $(B_0 : B_K)$ splits more than one R-set.

This heuristic has proven to be extraordinarily effective, and also quite fast. See Chapter 11.

9.2 Path-bridge heuristics

We propose three heuristics for the separation of PB-inequalities. They have in common that they first find a list of non-oriented paths which are then checked for violated PB-inequalities.

Algorithm 9.1 KC-heuristic based on cactus cycles

Input:
GRP-structure Γ, LP-solution $x^* \in \mathbb{Q}_+^{E(G)}$ satisfying all connectivity inequalities and cactus representation of all x^*-minimum cuts of $G^{\mathrm{s}}_{\mathcal{C}}$.

Output:
May find violated (switched) KC-inequalities.

```
1: For each cycle $\mathcal{C}$ of the cactus do
2:    Let $B_0, \dots, B_{l-1}$ be the sets of the circular partition defined by $\mathcal{C}$.
3:    For $i = 0, \dots, l-1$ do
4:       Using Algorithm 8.2, check for a violated (switched) KC-inequality using the path
         $(B_{i+1}, \dots, B_{i-1})$, where the indices are taken modulo $l$.
5:    End for
6: End for
```

Before we sketch the three algorithms, we introduce a (heuristic) variant of Algorithm 8.1, which is suitable for checking lists of paths which do not satisfy the condition (8.3) on page 108. There is no way to find weights for the edges in the complete graph with node set $\{s_i, t_i, s_j, t_j\}$, such that all seven possible positions of the two involved paths in the PB-configuration obtain the correct values for the minimum odd cut computation – the system of linear equations simply has no solution. The heuristic modification we chose is the following. For any choice of coefficients there can be a strictly positive deviation of the contribution of the edges in a odd cut from the true contribution of the position of the paths defined by the cut to the slack of the PB-inequality. Hence, from the perspective of the odd cut, a certain position of the two paths might have a positive (or negative) contribution to the slack of the inequality, but the true contribution can be different. We simply chose the coefficients of the six edges such that the maximum over all seven possible positions of this deviation is minimized.

When in the remainder of this section and the following section we refer to Algorithm 8.1, we mean this variant.

9.2.1 Path-finder

The first heuristic, which we called *path-finder,* searches for *simple* paths, i.e., each bead on each path consists of a single R-set. The algorithm also extends the basic search/check-approach by computing even cuts. By this we mean that the possibility that some paths in the list are not used in a PB-configuration, but are divided into two even parts each by $(A : Z)$ is, to a certain extent, allowed. However, some extra care needs to be taken to make this compatible with the modeling of the path by the addition of two nodes to the subgraph induced by the nodes which are not contained in any path. In particular, there is no way to find the weights for some of the edges of this graph if the path split by an even cut has three or more beads. Hence, for each path of length two in the list, we compute a minimum even (s,t)-cut in the graph induced by the nodes of the graph. In our description of the path-finder in Algorithm 9.2, we do not indulge in the details of this part, instead we focus on the way the paths are found.

The search for paths is guided by a number of parameters ϵ_i, which are small positive rationals. Algorithm 9.2 cannot make sure that all found inequalities are really violated: edges uv with $u \in B_j$ and $v \notin \bigcup_i B_i$ for a path $P = (B_1, \dots, B_n)$ and $1 < j < n$, may decrease the amount of violation. This is the rationale behind the parameter ϵ_2, which, if small, makes this less likely.

9.2.2 Circular partitions

This heuristic constructs a set of path-candidates by identifying segments of circular partitions from the cactus. We recall that the *block graph* of a graph H is a tree whose node set is the union of the set of blocks of H with the set of cut-nodes of H, and whose edges reflect the containment relation. The leaves of the block graph are blocks of H. Algorithm 9.3 computes the block graph

Algorithm 9.2 PB path-finder heuristic

Input:

GRP-structure Γ with bounds b on the variables, LP-solution $x^* \in \mathbb{Q}_+^{E(G)}$.

Output:

Violated PB-inequalities.

```
1: Let all R-sets be unmarked. Initialize the list of suitable paths, $\mathfrak{L}$, to be empty.
2: **While** there exist unmarked R-sets **do**
3:    Let $C$ be an unmarked R-set.
4:    Find a path consisting of unmarked R-sets by walking along edges of $G^{s}_{\mathfrak{C}}(x^*)$ with $x^*$-value
      at least $1 - \epsilon_1$, avoiding R-sets $D$ with $x^*(\partial(D)) \geq 2 + \epsilon_2$ as interior nodes and
      $x^*(\partial(D)) \geq 3 - \epsilon_3$ as end nodes.
5:    **If** such a path is found **then**
6:       Add it to the list of suitable paths $\mathfrak{L}$, mark all its nodes.
7:    **End if**
8: **End while**
9: **For** each path $P$ in the list $\mathfrak{L}$ **do**
10:   **If** $P = (B_1, B_2)$ has length two **then**
11:      Construct the subgraph $H_P$ of $G$ induced by the nodes in $P$. Add two extra nodes $s, t$
         to $H_P$, and for each edge $uv \in \partial(B)_1 \setminus \partial(B)_2$ with $v \in B_1$ add an edge $sv$ with weight
         two. The end nodes of the edges in $\partial(B)_1 \setminus \partial(B)_2$ are connected to $t$ in the same way.
         Make the nodes $s$ and $t$ odd. All other nodes get the parity $\mathbf{t}$.
12:      Compute a minimum odd $(s,t)$-cut in $H_P$, and let $\lambda^0(P)$ denote its value.
13:   **Else**
14:      Set $\lambda^0(P) := \infty$.
15:   **End if**
16: **End for**
17: Construct the subgraph $H$ of $G$ induced by the nodes which are not contained in a path.
18: **For** each path $P = (B_1, \ldots, B_n)$ **do**
19:   Add two nodes $s_P$, $t_P$ to $H$. Connect them with an edge which gets the weights
      $c^0_{s_P t_P} := \lambda^0(P)$ and $c^1_{s_P t_P} := -1 + \frac{1}{n-1} x(\partial(\bigcup) B_i) + \sum_{1 \leq i < j \leq n} x(B_i : B_j)$.
20:   For all $v \in V(H)$, if there are edges $uv$ in $G$ with $u \in B_1$, add to $H$ an edge $s_P v$ with
      weight $c^0_{s_P v} := x^*(B_1 : v)$ and $c^1_{s_P v} := \infty$. Similarly for $B_n$ and $t_P$.
21: **End for**
22: If there are edges $uu'$ with $u \in B_1$ for a path $P = (B_1, \ldots, B_n)$ and $v \in B'_1$ for a path
    $P' = (B'_1, \ldots, B'_{n'})$, add to $H$ an edge $s_P s_{P'}$ with weights $c^0_{s_P s_{P'}} := x^*(B_1 : B'_1)$ and
    $c^1_{s_P s_{P'}} := \infty$. Similarly for pairs $B_1, B'_{n'}$ and $B_n, B'_{n'}$.
23: For each edge $e$ in $G(x^*)$ with both end nodes in $V(H)$, the edge $e \in E(H)$ is given the
    weights $c^0_e := x^*_e$ and $c^1_e := b_e - x^*_e$.
24: The parities of the nodes $s_P$, $t_P$ are odd; all other nodes of $H$ get the parity $\mathbf{t}$.
25: Compute a minimum blossom in $H$. For each blossom with value strictly less than one,
    construct the corresponding PB-inequality.
```

Algorithm 9.3 PB-heuristic based on cactus cycles

```
Input:
    GRP-structure $\Gamma$, LP-solution $x^* \in \mathbb{Q}_+^{E(G)}$ satisfying all connectivity inequalities and cactus
    representation of all $x^*$-minimum cuts of $G^{\mathrm{s}}_{\mathbb{C}}$.
Output:
    May find violated (switched) PB-inequalities.
1:  $\mathcal{L}$ is a list of paths, initialized to be empty.
2:  Compute the block graph $T$ of the cactus graph $\mathcal{K}$.
3:  Let all cut-nodes of $\mathcal{K}$ be unmarked.
4:  While $T$ is not empty do
5:      Select a leaf of $T$, preferring leaves which represent cut-edges of $\mathcal{K}$. (If $T$ consists of a
        single node, we consider this node as a leaf.)
6:      If the selected leaf represents a cycle of the cactus then
7:          Unless $T$ consists of only one node, let $B'$ denote the cactus cut-node adjacent to the
            leaf in $T$.
8:          Identify all inclusion maximal segments of the cycle not containing $B'$ or a marked
            cactus node. For every segment whose length is at least two, store the path it defines in
            $\mathcal{L}$. If any such exists, mark the cut-node $B'$.
9:      End if
10:     Unless $T$ consists of only one node, if $B'$ is not marked, check if one of the cactus nodes of
        the block is marked, and if so, mark $B'$.
11:     Delete the selected leaf from $T$. If by the deletion of the leaf a cut-node of the cactus $\mathcal{K}$
        has become a leaf of $T$, delete this leaf too.
12: End while
13: Using Algorithm 8.1, check the list of paths $\mathcal{L}$.
```

T of the cactus graph, and modifies it, maintaining as an invariant of the loop in steps 4 to 12 that the leaves are blocks of the cactus.

9.2.3 Cactus without cut-nodes

Finally, we introduce the most expensive heuristic. The basic idea is the following. In $G^{\mathrm{s}}_{\mathbb{C}}(x^*)$ delete all nodes which are contained in cut-nodes of the cactus, and shrink the contents of each cactus node which is not a cut-node. This leaves us with a graph, in which every node has the property that x^*-values of its incident edges sum up to at most two. This concept combines the circular-partition strategy of the Algorithm 9.3 with the path-finder strategy of Algorithm 9.3. However, we avoid some of the disadvantages of the latter, for example that each node consists of only one R-set. It turns out that this heuristic finds important PB-structures, which are "hidden" in the cactus.

The way the paths are formed by Algorithm 9.4 differs from the approach of Algorithm 9.3. Let H denote the graph obtained from $G^{\mathrm{s}}_{\mathbb{C}}(x^*)$ in the way described above, and let δ_0 denote the minimum value such that deleting every edge of H with x^*-value less than δ_0 results in a graph with maximum degree two. Now select a δ in the interval $[\delta_0, 1]$, and delete all edges with x^*-value less than δ, and, further, from every circle the edge with minimal x^*-value. This results in a set of paths of H. These paths are the candidates which are checked for violated KC- and PB-inequalities. The process is repeated for different values $\delta \in [\delta_0, 1]$.

9.3 A note on HC-separation

Honeycomb inequalities are "more difficult" to separate than PBs. Firstly, a result resembling Proposition 8.3.6 for HC-inequalities with $L = 1$ (see 2.1.2 for the notation) would involve an algorithm to partition the node set of a graph into n_1 odd subsets, such that a sum of the form

Algorithm 9.4 PB-heuristic based on cactus cut-nodes

```
Input:
    GRP-structure $\mathit{\Gamma}$, LP-solution $x^* \in \mathbb{Q}_+^{E(G)}$ satisfying all connectivity inequalities and cactus
    representation of all $x^*$-minimum cuts of $G^{\mathrm{s}}_{\mathfrak{C}}$.
Output:
    May find violated (switched) PB-inequalities.
 1: Construct the graph $H$ whose node set is the set of all non-cut-nodes of the cactus, with an
    edge between two cactus nodes $B'$ if $x^*(B : B') > 0$. The edge weight $w_{BB'}$ is the value
    $x^*(B : B')$.
 2: Compute the minimum value $\delta_0$ with the property that every node in $H$ has at most two
    edges with weight greater than or equal to $\delta_0$.
 3: Repeat
 4:   Select $\delta \in [\delta_0, 1]$.
 5:   Construct the graph $H_\delta$ which results from $H$ by deleting all edges with weight strictly
      less than $\delta$ are deleted.
 6:   In every circle of $H_\delta$ delete the edge with smallest weight.
 7:   Let $\mathcal{L}$ denote the list of paths in $G$ defined by the set of all paths of $H_\delta$.
 8:   Check each path in $\mathcal{L}$ for a violated (switched) KC-inequality, using Algorithm 8.2.
 9:   Check the list of paths $\mathcal{L}$ for violated (switched) PB-inequalities using Algorithm 8.1.
10: Until a termination condition is satisfied.
```

$\sum_{i \neq j} c_{ij} w(X_i : X_j)$ is minimized. Apparently, this cannot be done by min-odd-cut techniques. Secondly, if based on a cactus, there is no promising way to connect the neighboring beads of a cactus node to the sets X_i such that the conditions in the definition of HCs are satisfied. The best approach to this problem seems to be to start with a partition B_i^1, $i = 1, \ldots, n_1$, and then to try to construct the neighboring beads. This is just the approach proposed in by Corberán et al. in [CLS01]. In our Branch-and-Cut implementation, we use their code.

Part III

Branch & Cut

Chapter 10

Strategies inside Branch-and-Cut

In this chapter we discuss how the theoretical knowledge about the General Routing Problem guided our design of a computer code to solve the GRP by Branch-and-Cut. Before we proceed, a note on the support graphs $G(x^*)$ and $G_{\mathcal{C}}(x^*)$ is due. They are always *simple* graphs, but to each edge, there may be associated a set of variables. We say that the variable *belongs to* the edge. In the case of $G(x^*)$, this may be caused by shrinking operations, or by the presence of duplicate variables for each edge of a Ghiani-Laporte tree (see Section 2.2). Technically, the sets of variables belonging to edges are visible to the separation modules.

10.1 Core iteration, feasibility test, and first separation routines

In this section we explain the core iteration of the B&C process, namely how the LP-solution x^* is treated after the solution of a Linear Programming relaxation. It is first checked for feasibility, which amounts to a quick separation of connectivity and R-odd cut inequalities. If x^* is integer and no violated connectivity or R-odd cut inequalities have been found, it is necessary to check the parity of each node.

Algorithm 10.1 displays a core iteration. The reader will notice the extensive test for violated connectivity inequalities, as displayed in Algorithm 10.2. The *exact* separation routine, namely the min-cut computation, is even *repeated* a couple of times after a random permutation of the nodes, to produce more violated connectivity inequalities. This is to ensure, as early as possible in the B&C process, that all connectivity inequalities are satisfied and the separation heuristics based on the cactus representation of all minimum cuts can be used. However, some of the steps of Algorithm 10.2 are omitted if the edge weights of $G_{\mathcal{C}}(x^*)$, i.e., the values $x^*(C : D)$ for each pair of adjacent R-sets C, D, are integer. The reader might also notice that comparatively few energy is spent on the first search for violated R-odd cut inequalities in Algorithm 10.3. The rationale behind this is the fact that adding too many R-odd cut inequalities has the side effect of destroying nice and usable structures,[1] for example many or big cycles in the cactus. In fact we even refrain from executing steps 2 to 6 in every cutting plane iteration.

10.1.1 Block decomposition

Before further time-consuming separation routines are started, we perform a block decomposition, see Proposition 6.1.1. In this, we follow the approach of [CLS01] for the separation of simple 2-PBs, who propose to add to $G(x^*)$, prior to computing the blocks, all required edges, and then computing a block decomposition.

We modify the standard way of performing a block decomposition in two ways, with the aim of increasing the number of blocks. Prior to invoking the block decomposition algorithm, we first

[1]The negative effect of the R-odd cut inequalities was also observed by A. Corberán [Cor04].

Algorithm 10.1 Core iteration of the B&C algorithm for the GRP

```
Input:
    GRP-structure $\Gamma$, LP-solution $x^*$.
 1: Check if $x^*$ is integer.
 2: Do a thorough global search for violated connectivity inequalities (see Algorithm 10.2).
 3: Do a quick-and-dirty search for violated odd-cut inequalities (see Algorithm 10.3).
 4: If $x^*$ is integer and no violated inequality was found then
 5:    Test the parity of each node.
 6: End if
 7: If $x^*$ is a feasible solution then
 8:    Store the feasible solution and prune this branch of the branch-and-bound tree.
 9: Else if $G_{\mathcal{C}}(x^*)$ is connected then
10:    Invoke a heuristic to produce a feasible GRP solution from $x^*$ (see Section 10.2). Most of
       the times the quick Algorithm 10.5 is used, occasionally Algorithm 10.4.
11:    Do a block decomposition (see 10.1.1).
12:    For each block do
13:       Test if the restricted $x^*$-values represent a feasible GRP-solution on the block.
14:       If that is not the case, invoke more subtle separation procedures (see Section 10.3).
15:    End for
16:    If no violated inequality was found then
17:       Try to tighten the upper bound constraints (see Section 10.4), or initiate a branching
          step (see Section 10.5).
18:    End if
19: End if
```

Algorithm 10.2 Global separation of connectivity inequalities

```
Input: Graph $G_{\mathcal{C}}(x^*)$.
Output: Violated connectivity inequalities (many of them), if any exist.
 1: Test for connected components of $G_{\mathcal{C}}(x^*)$.
 2: Test each R-set $C$ for $x^*(\partial(C)) \geq 2$.
 3: Test each pair $\{C, D\}$ of adjacent R-sets for $x^*(\partial(C \cup D)) \geq 2$.
 4: Check the connected components of the graph $G_{\mathcal{C}}(x^*) \setminus E_0$, where $E_0 := \{e \mid x^*_e < 1/3\}$.
 5: Find the smallest value $\varepsilon$ such that the sum of all R-external edges with value $\leq \varepsilon$ is greater
    than or equal to 3, and check the connected components of the graph $G_{\mathcal{C}}(x^*) \setminus E_0$, where
    $E_0 := \{e \mid x^*_e < \varepsilon\}$.
 6: For each connected component of $G_{\mathcal{C}}(x^*)$ do
 7:    Compute a minimum cut using the Hao-Orlin algorithm [HO94]; in each iteration (i.e.,
       after each min $(S, t)$-cut computation, see 0.4.1) of the the Hao-Orlin procedure, store a
       violated connectivity inequality if $\lambda(S, t) < 2$.
 8:    Repeat the Hao-Orlin min-cut computation after a random permutation of the nodes, until
       no new inequalities are found, or at most $\max(1, \log(|n_p|) - 2)$ times, where $n_p$ is the
       number of nodes in the connected component.
 9: End for
```

Algorithm 10.3 Quick-and-dirty search for violated R-odd cut inequalities

```
Input: Graph $G(x^*)$, parities.
Output: Possibly some violated R-odd cut inequalities.
1: Test for odd connected components of $G(x^*)$.
2: Test each odd node for $x^*(\partial(u)) \geq 1$.
3: Test each edge $\{u,v\}$ with $\mathbf{t}(u) \neq \mathbf{t}(v)$ for $x^*(\partial(\{)u,v\}) \geq 1$.
4: Check the connected components of the graph $G(x^*) \setminus E_0$, where $E_0 := \{e \mid x^*_e < 1/3\}$.
5: Find the smallest value $\varepsilon$ such that the sum of all edges with value $\leq \varepsilon$ is greater than or
   equal to $3/2$.
6: Check the connected components of the graph $G(x^*) \setminus E_0$, where $E_0 := \{e \mid x^*_e < \varepsilon\}$.
```

delete all R-internal edges e with $x^*_e \geq 1$ and then add a minimal set of edges such that each R-set is connected (instead of adding all required edges). This is justified by Proposition 6.2.1. After the block decomposition is computed, we restore the deleted edges (with their proper values) to the blocks.

The feasibility test is repeated locally on each block, which allows to avoid invoking time consuming separation routines on blocks which represent feasible solutions, a situation which frequently occurs.

10.2 Heuristics for feasible solutions

When an LP is computed in the Branch-and-Cut process, after testing the solution for feasibility and before separation, we invoke an algorithm to produce a feasible solution. The idea is to use "information" contained in the LP-solution x^* to find a good upper bound. We use two algorithms. One very simple algorithm runs in time $O(m \log n)$, another has a worst case running time of $O(n^3)$, or, to be precise, the time to compute a minimum weight perfect matching in a complete graph with $|V(G)|$ nodes. The idea of this algorithm goes back to a $3/2$-approximation heuristic for the RPP described in [BCCM85, Jan93]. The algorithm computes an x^*-maximum spanning tree in $G^{\mathrm{m}}_{\mathbb{C}}(x^*)$. However, this part of the algorithm is randomized, so that a tree can be found, which is not maximum. Then a minimum T-join is computed with respect to the cost data of the instance. This guarantees that all nodes obtain the correct parity in the solution. The minimum T-join computation is quite time consuming. A set of trees for which a T-join has already computed is stored, in order to avoid repeatedly computing the same T-join. The shortest path computations which are necessary to reduce the T-join problem to a matching problem are also only performed once during the B&C procedure.

The T-join based heuristic is quite expensive computationally. The second algorithm we propose runs in time $n \log n$. It starts by computing a maximum spanning tree in $G^{\mathrm{m}}_{\mathbb{C}}(x^*)$, but to make sure that every node obtains the right parity in the solution, it simply computes a second, edge disjoint spanning tree, which is minimum with respect to the edge costs of the instance. In this second tree, the unique T-join can be found in linear time. Algorithm 10.5 displays this approach. It works very fast, but it does not improve the upper bound very often.

In fact, our algorithms are a bit more elaborate than Algorithms 10.4 and 10.5, since they allow to honor upper bounds on the variables.

10.3 Minimum odd cuts and blossoms in practice

The separation heuristics described in Chapter 9 and the (switched) simple 2-PB separation of Section 8.2 all use either minimum odd cuts or blossom minimization, possibly with the restriction of separating a pair of nodes s, t. In this section, we explain the implementation of the modules which perform these tasks.

Algorithm 10.4 GRP upper bound heuristic based on T-join

```
Input:
    GRP-structure $\Gamma$, and LP-solution $x^* \in \mathbb{Q}_+^{E(G)}$.
Output:
    May produce a feasible solution $x$ for $\Gamma$.
1: If this is the first call to this algorithm during the B&C procedure, initialize the set $\mathcal{Y}$ of
   spanning trees which have already been treated to $\emptyset$.
2: Create a random cost vector $w$ for $G_{\mathbb{C}}^{\mathrm{m}}$ such that $|x_e^* - w_e| \leq 1/10$ for all $e \in E(G_{\mathbb{C}}^{\mathrm{m}})$.
3: Compute a $w$-minimum spanning tree in $G_{\mathbb{C}}^{\mathrm{m}}$. Denote the characteristic vector of its edge set
   by $y$.
4: If $y \notin \mathcal{Y}$ then
5:    Let $\mathcal{Y} := \mathcal{Y} \cup \{y\}$.
6:    Let $T$ be the set of all nodes $u$ of $G$ for which $\mathbf{t}(u) + y(\partial(u))$ is odd.
7:    Compute a minimum $T$-join in $G$, denote its characteristic vector by $z$.
8:    Return $x := y + z$.
9: End if
```

Algorithm 10.5 GRP upper bound heuristic “Double-Tree”

```
Input:
    GRP-structure $\Gamma$, and LP-solution $x^* \in \mathbb{Q}_+^{E(G)}$.
Output:
    Feasible solution $x$ for $\Gamma$.
1: Create a random cost vector $w$ for $G_{\mathbb{C}}^{\mathrm{m}}$ such that $|x_e^* - w_e| \leq 1/10$ for all $e \in E(G_{\mathbb{C}}^{\mathrm{m}})$.
2: Compute a $w$-minimum spanning tree in $G_{\mathbb{C}}^{\mathrm{m}}$. Denote the characteristic vector of its edge set
   by $y$.
3: Compute a $c$-minimum spanning tree $H$ in $G$ where $c$ denotes the cost vector of the GRP
   instance.
4: Let $T$ be the set of all nodes $u$ of $G$ for which $\mathbf{t}(u) + y(\partial(u))$ is odd.
5: Find the unique $T$-join in $H$, denote its characteristic vector by $z$.
6: Return $x := y + z$.
```

Algorithm 10.6 Blossom minimization module

Input: Graph G, set $T \subseteq V(G)$, and weights $c, c' \colon E(G) \to \mathbb{Q}_+$.
Output: Blossoms (U, F) with value strictly less than 1, if one exists.

```
1: Set $w := \min(c, c')$.
2: Compute connected components of the spanning subgraph of $G$ with edge set and check
   them for small blossoms.
3: For all connected components $H$ do
4:   If $V(H)$ is an odd set then
5:     Store the blossom.
6:     Check nodes of $H$ for small blossoms.
7:     Using the Hao-Orlin algorithm [HO94], compute a minimum cut in $H$. In the process,
       check all computed minimum $(S, t)$-cuts (see 0.4.1) for small blossoms.
8:   Else /* $H$ is even */
9:     Compute a block decomposition of $H$.
10:    For all blocks $H'$ of $H$ do
11:      Examine the cut-nodes of $H$ which are contained in $H'$ for small blossoms.
12:      Invoke a variant of Algorithm 7.5, which examines all edges of the cut-tree and store
         the blossom if its value is $< 1$.
13:    End for
14:  End if
15: End for
```

A folklore rule of thumb in the design of B&C algorithms is to produce many violated inequalities in each call of a separation algorithm. This is achieved by adopting the modules for computing minimum odd cuts, minimum odd (s,t)-cuts and minimum blossoms.

10.3.1 The minimum odd cut module

The module for the computation of a minimum T-odd cut in a graph H starts exactly as Algorithm 10.3 and then invokes a variant of Algorithm 7.4, which stores every odd cut with value strictly less than a given upper bound (usually 1). It operates in three phases. The computation can be interrupted, if (enough) odd cuts with small value have been found.

I. Compute connected components, check if they are odd.

II. In each even connected component, use the heuristics of steps 4 to 6 of Algorithm 10.3.

III. In each even component, invoke Algorithm 7.4.

10.3.2 Adapting the blossom minimization algorithm

We use blossom minimization for the separation of (switched) simple 2-PBs in 8.2.1 and for the path-bridge heuristic in Algorithm 9.2. In particular for the latter application, we want to find many blossoms with small value, in the hope of finding many different path bridge inequalities. To meet this requirement we have designed a blossom minimization module which is displayed as Algorithm 10.6. It uses decompositions of H into connected components and blocks. To obtain a large number of small blossoms, odd connected components are treated by computing minimum cuts.

10.3.3 Computation of minimum odd (s,t)-cuts and (s,t)-blossoms

Many of our algorithms, including 8.1, 8.2, 9.1, 9.2, 9.3, and 9.4, require to find a odd (s,t)-cut (or (s,t)-blossom) with value strictly less than one. However, the computation of a minimum odd (s,t)-cut involves the computation of $O(n)$ maximum flows, plus the associated DAG-representations of all minimum (x, y)-cuts. In practice, this is considerably more time consuming than just computing minimum (x, y)-cuts.[2] Further, the max-flows cannot be computed on shrunk graphs as in the Gomory-Hu cut-tree procedure or Algorithm 7.4. Even the number of required max-flow and DAG computations in case of the minimum odd (s, t)-cuts is in practice a lot higher than for a minimum odd cut computation using Algorithm 7.4 (see the following chapter, in particular Table 11.3).

The good news is that, in the application within a B&C algorithm for the GRP, if there exists no violated R-odd cut (or cocircuit) inequality, then the only odd cut with value strictly less than one must be the one separating the two "artificial" nodes s, t in the auxiliary graphs constructed by Algorithms 8.2 and 8.1. However, we do not want to wait so many iterations of the cutting-plane algorithm until no violated R-odd cut inequalities exist; and for the cocircuit inequalities, this is a really time-consuming task. Moreover, as we will will describe in the next section, it is common experience among those working on cutting-plane algorithms for the GRP that the presence R-odd cut inequalities and, to an even greater extend, cocircuit inequalities in the LP make it more difficult for other separation heuristics to find violated inequalities.

Thus, the way we actually search for a small odd (s, t)-cut (or small (s, t)-blossoms) in the separation of variants of path bridge and KC-inequalities, regardless of whether violated R-odd cuts exist or not, is to compute as many small odd cuts (or blossoms) in the graph as possible, in the hope that some of them might be (s, t)-cuts. In practice, this works very well. Violated inequalities are very often found in the phases I and II of the minimum odd cut module, because the violated R-odd cut inequalities which could be identified this way are separated at the beginning of each core iteration, see Algorithms 10.1 and 10.3.

In order to make it more likely that odd (s, t)-cuts are found, our minimum odd (s, t)-cut module treats even connected components of the graph H depending on whether they contain one, both, or none of the nodes s and t, and whether s and t are contained in two distinct (even) connected components. For example, any odd cut in a component containing either s or t can be used to produce an odd (s, t)-cut of the same value. If the nodes s and t are contained in distinct (even) connected components, then an odd cut in a component not containing s or t can be used to form an odd (s, t)-cut with the same value.

10.4 Selecting inequalities and variable bounds

Obtaining good dual bounds by a cutting-plane algorithm based on heuristic separation routines is heavily subject to random effects. The inclusion of a quick exact separation routine for a simple class of inequalities into each cutting-plane iteration does not guarantee better dual bounds. On the contrary—it is a common experience that it has the opposite effect. Furthermore, after adding a new separation routine to a cutting-plane code, the dual bounds will improve for some instances, and for some they will get worse. In our work, we have not undertaken to fine-tune a list of separation routines to produce best average gap over a small set of mildly difficult instances. We grant, though, that this type of fine tuning can have a considerable effect. For example, we were unable to reproduce the results of [CLS01] even with the source code of the separation routines which were used in that paper. However, we might add that our lower bounds, based on our new classes of inequalities, an improved understanding of the LP-solutions, our own separation routines and some other techniques which we will describe in this section, considerably improve on those in [CLS01, GL00], even without fine tuning.

10.4.1 Gather ye rosebuds while ye may

It has been observed [CLS01] that the presence of the upper bounds $x_e \leq 1$ and, more strikingly, the adding of cocircuit inequalities to the LP have a devastating effect on usable structure of the LP-solution. If cocircuit inequalities are separated early in a cutting-plane algorithm for the GRP, then, on difficult instances, the final lower bound will be *considerably* worse than if this class of inequalities had been postponed to the point where none of the "larger" structures, namely PBs,

[2] K. M. Wenger has made computational tests in this issue and communicated the result to us.

KCs, and HCs, can be found. However, it has also been observed that cocircuit inequalities improve the lower bound: for many of the GRP instances proposed in [CLS01], it is easy to achieve the optimal lower bound or a very small gap using only connectivity and cocircuit inequalities. R-odd cut inequalities have the same tendency as cocircuit inequalities, though not as strong. Before we used the strategy which we will describe below, the lower bounds for two instances widely-known to be difficult, namely GRP04 and GRP10 *increased* considerably if the separation of R-odd constraints was turned off completely until the point when *no* other violated inequalities were found. (Needless to say that the lower bounds for many other instance deteriorated dramatically.)

Not only R-odd cut and cocircuit inequalities are dangerous. Theoretical results on the strength of inequalities indicate that configurations with long paths have the potential to increase the lower bounds more than those with short paths, like for example 2-PBs [Goe95, Let04]. It has also been observed that adding 2-regular PB-inequalities to the LP makes it more difficult to find PBs with long paths later in the cutting-plane process. The separation of 2-PBs which are not simple is quite difficult though.

The conclusion of the above is that there is a category of inequalities which are hazardous to the separation of other inequalities. The question is: how can all inequalities which are found by delicate heuristics be used in the cutting plane process, but at the same time destroying as little structure as possible. We propose the following strategy for the problem. As long as the LP-solution has easily recognizable structure, we invest much time to find as many path-inequalities as possible. The *list of paths* of each of these inequalities are stored. Then, for every inequality, we make a guess as to whether it is dangerous to the structure of the next LP-solution. If we decide to the positive, we simply discard the inequality, but we keep the stored list of paths. At a later point in the B&C-algorithm, for example just before branching or after the B&B-tree exceeds a certain depth, we scan through the set of stored list of paths, and try to find a violated PB-inequality which uses a subset of the list of paths. For this last step, we invoke a min-odd-cut based algorithm in the fashion of Algorithms 9.3 and 9.4.

Storing the set of all list of paths is also used to improve pool separation. Inequalities which have been found to be violated at some point in the of the B&C process are checked from time to time if they are again violated. Pool separation checks each inequality, while our techniques allow to search a whole “neighborhood” of the inequality. This is inspired by the tightening and teething ideas of [ABCC03, ABCC].

Storing the paths-lists independently from the inequalities has the additional advantage of reducing the memory requirement of the B&C-program, as there are usually many violated PB-inequalities with the same list of paths.

10.4.2 Upper bounds on the variables and the Ghiani-Laporte tree

For the upper bounds of the variables [CLS01] propose the strategy to not add the upper bound inequalities until late in the cutting-plane process. The upper bounds $\mathbf{b}^T$ depend on the Ghiani-Laporte tree T. The question how the selection of the tree, if there exist more than one, influences the cutting-plane progress, and how the best tree can be selected was raised by A. Corberán. Table 10.1 shows the lower bounds of the relaxation using only connectivity and switched 2-PB inequalities can change for different trees. For some trees, the lower bound is extremely good or optimal, for others, it is far worse.

We propose the following strategy. We start with no “dangerous” upper bounds, i.e., the upper bound is two or ∞ on the R-external edges, and one or ∞ on the R-internal edges. At a certain point in the progress of the B&C-algorithm, for example just before branching or if the B&B-tree reaches a certain height, we invoke the cocircuit separation, pretending that we had added the upper bound $x_e \leq 1$ for *all* edges e, which is not feasible for the GRP, of course. Then, for increasing values of $k = 1, 2, \ldots$, we consider all cocircuit inequalities $x(\partial(W) \setminus F) - x(F) \geq 1 - |F|$ among all we found, for which the set F consists of k edges. We then try to find a set of edges E_1, such that $E(G^{\mathrm{m}}_{\mathcal{C}}) \setminus E_1$ still contains a minimum spanning tree for $G^{\mathrm{m}}_{\mathcal{C}}$, and such that some violated blossom inequalities will be found when these inequalities are present. For those edges which might or might not be in a minimum spanning tree of $G^{\mathrm{m}}_{\mathcal{C}}$, we make the upper bound

instance	lower bounds for different trees					opt.
GRP04	4807	4817	4840	4845	4855	5186
GRP08	6800	6814				6814
GRP09	4494	4498	4506			4506
GRP10	4711	4721	4744	4749	4759	5122

Table 10.1: How the choice of the Ghiani-Laporte tree influences the lower bound for a relaxation

inequalities local to the current subproblem of the branch-&-bund procedure, and its predecessors in the branch-&-bound tree. This of course implies that the blossom and switched PB-inequalities which use such edges in their sets F are also local to these subproblems.

10.5 Branching

For branching we use two different strategies, one based on parity the other on connectivity. For the parity part, we search for even cuts $(U, V(G) \setminus U)$ with $x^*(\partial(U)) \approx \delta$, where δ is an odd integer. Then we take the branches $x(\partial(U)) \leq \delta - 1$ and $x(\partial(U)) \geq \delta + 1$. This approach has been in use for a while for the TSP. We prefer sets U which are unions of R-sets. To search for these cuts, we use the blossom minimization algorithm, which is a new approach.

For the connectivity based branching, we rely on Proposition 6.2.3. Starting with $H := G^{\mathrm{m}}_{\mathfrak{C}}(x^*)$, for every variable with value greater than or equal to one which belongs to an $e \in E(H)$, we contract the edge e in H. Then we consider all variables belonging to fractional edges of the resulting multigraph. In other words, we consider the set of variables belonging to edges e of G which are R-external and have their end nodes in different connected components of the spanning subgraph of G with edge set $E_{\mathrm{int}} \cup \{e \in E(G) \mid x_e \geq 1\}$.

After building a set of candidate branching decisions, we use the well-known *strong branching* method to select one of them. This means that, for each branch in each of the candidates, the corresponding inequality is tentatively added to the LP, a number of dual simplex iterations are performed (actually the LP is solved), and the resulting dual bound is stored. Then the candidate is selected for which the worst bound is best.

Chapter 11

Performance of separation algorithms

In this chapter we present computational results for the separation algorithms in terms of running times and number of inequalities found. Before we do that a word on the computer platform. We used the GNU C/C++ compile *gcc* in Version 3.3. We heavily use the Standard C++ Library. For the graph data structures, we rely on our own code framework, called *Heidegger*. All running times refer to Intel Xeon PCs with 2.8 GHz running LINUX. All running times are given in seconds.

11.1 The core minimum odd-cut algorithm

We start with comparing the data of our implementation of Algorithm 7.4, which we called HeMOC *(Heidelberg Min-Odd-Cut)* to three alternative algorithms, namely a min-odd-cut program based on the cut-tree algorithm by Gusfield [Gus90] from the Konrad-Zuse Zentrum für Informationstechnik (ZIB) in Berlin implemented by G. Skorobohatyj, which we refer to as"ZIB", our own implementation of the core of Algorithm 7.4, i.e., without shrinking, which we refer to as "Core", and the Gomory-Hu cut-tree module which is part of Concorde, Version 99.12.15, the TSP software by Applegate et al., which we refer to as "Concorde". We note that the cut-tree code of this version is identical to that of the latest Version 03.12.19.

We implemented the Algorithm 7.4 in C++, using parts of the code described in the paper [JRT00], in particular the graph data structure, shrinking and the MCAP function. We used the minimum (s,t)-cut code which is part of Concorde, Version 99.12.15, which is identical to that of the latest Version 03.12.19. The fact that "HeMOC", "Core", and "Concorde" use the same minimum (s,t)-cut code facilitates a comparison. All the codes are optimized to find only *one* minimum odd cut. For the comparison, we use practical instances which arouse in the separation of R-odd cut inequalities in our GRP-solver. The sizes are between 300 and ca. 3400 nodes. The results are displayed in the tables 11.1–11.4. The first column of all tables denotes the number of nodes, averaged over a set of instances of approximately the same size. The average number of odd nodes in the set is denoted by $|T|$.

We start with some information about the effectiveness of the shrinking mechanisms of Algorithm 7.4. Table 11.1 displays how many calls to the MCAP-procedure were successful: the column "# MCAP total" contains the average total number of calls, while the column "# MCAP successful" contains the average number of calls which resulted in an updated value of ϕ or a possibility to shrink. It can be seen that there is no great risk to lose much time in unfruitful MCAP calls. Table 11.1 compares the number of max-flow computations (column "# max flow HeMOC") to the total number of calls to the MCAP-procedure. Table 11.3 compares the number of max-flow computations of Algorithm 7.4 when shrinking is enabled (HeMOC) or disabled (Core). The averaged absolute values are displayed as well as the percentage of max-flow computations relative to the number of max-flows which need to be computed by the Padberg-Rao algorithm

n	$\|T\|$	m	# MCAP	# MCAP
			total	successful
317.27	178.35	386.21	3.71	3.41
973.22	512.51	1152.01	3.32	3.06
1475.68	584.55	1613.56	19.33	18.65
1788.17	542.32	1826.20	30.81	27.99
2180.39	756.91	2328.54	22.39	19.82
2497.23	1070.16	2811.82	15.91	13.46
3023.34	1205.04	3446.06	14.46	11.58
3325.80	866.41	3404.10	40.53	34.52

Table 11.1: Numbers of successful MCAP computations of Algorithm 7.4 (HeMOC)

n	$\|T\|$	m	# MCAP	# max-flow
			total	HeMOC
317.27	178.35	386.21	3.71	0.02
973.22	512.51	1152.01	3.32	0.00
1475.68	584.55	1613.56	19.33	0.02
1788.17	542.32	1826.20	30.81	1.37
2180.39	756.91	2328.54	22.39	1.20
2497.23	1070.16	2811.82	15.91	1.45
3023.34	1205.04	3446.06	14.46	1.99
3325.80	866.41	3404.10	40.53	2.71

Table 11.2: Number of MCAPs vs. number of max-flows of Algorithm 7.4 (HeMOC)

n	$\|T\|$	# max-flow	% max-flow	# max-flow	% max-flow
		Core	Core	HeMOC	HeMOC
317.27	178.35	124.70	70.31%	0.02	0.01%
973.22	512.51	355.17	69.43%	0.00	0.00%
1475.68	584.55	398.77	68.33%	0.02	0.00%
1788.17	542.32	356.39	65.83%	1.37	0.25%
2180.39	756.91	506.03	66.94%	1.20	0.16%
2497.23	1070.16	728.40	68.12%	1.45	0.13%
3023.34	1205.04	815.32	67.71%	1.99	0.16%
3325.80	866.41	567.21	65.54%	2.71	0.31%

Table 11.3: Numbers of max-flow computations of the Min-Odd-Cut core (HeMOC)

n	$\|T\|$	HeMOC	Core	ZIB	Concorde
317.27	178.35	0.00	0.02	0.07	0.01
973.22	512.51	0.00	0.15	0.55	0.05
1475.68	584.55	0.01	0.31	1.15	0.15
1788.17	542.32	0.01	0.55	1.45	0.37
2180.39	756.91	0.01	0.70	2.28	0.41
2497.23	1070.16	0.01	0.78	3.06	0.40
3023.34	1205.04	0.02	0.97	3.89	0.43
3325.80	866.41	0.02	1.62	4.47	0.90

Table 11.4: Running times of the different minimum odd cut algorithms

[PR82], namely $|T| - 1$.

Finally we give the running times for the four minimum odd cut algorithms in Table 11.4. As in the other tables, the numbers are averaged over the instances in the sets of similar size.

The conclusion is that the Gusfield based algorithm "ZIB" is not competitive, the core of Algorithm 7.4, without shrinking, performs surprisingly well, and that the only true competitor is the Gomory-Hu based algorithm which is part of Concorde.

We have made additional tests comparing "HeMOC" with "Concorde" on bigger instances. For them, we glued together graphs taken from the other tests, to obtain graphs for which the number of nodes was in either the range $2000, \dots, 4000$, or $10000, \dots, 18000$. We randomly added edges until the minimum degrees exceeded a certain value. The costs of the original edges were slightly perturbed, the costs of the new edges were randomly chosen. We tested the two algorithms on ca. 540 instances with average degree between 3.5 and 3.7, see Fig. A.1, ca. 540 instances with average degree between 4.1 and 4.3, see Fig. A.2, and ca. 830 instances with average degree between 6.3 and 6.5. The odd nodes were chosen randomly with probability $1/2$. In the three figures, the number of nodes is on the horizontal and the running times in seconds is on the vertical. All points in the area $n \geq 4000$, $t \geq 50s$ belong to Concorde. We also tried instances with average degree between 11.5 and 11.7, but Concorde could not solve such instances when $n \geq 4000$ (while HeMOC showed no problems with the sizes $n = 15000, \dots, 18000$. We would like to note that we did not undertake the effort to fine tune our implementation of HeMOC to optimize its performance. Hence, there can be no doubt that it is the introduction of Algorithm 7.4 which improves the practical solvability of the minimum odd cut problem.

11.2 The blossom minimization core

We now compare our implementation of the blossom minimization Algorithm 7.5 with the implementation of the Padberg-Rao algorithm which is part of Concorde, Version 99.12.15. As the theoretical improvement of the running time is a factor of $O(n/m)$, and $m \in O(n)$ holds for the separation problems where we apply the algorithms, we may hope for a constant factor improvement.

Our implementation of Algorithm 7.5 is in C++ and relies on its standard library. For the computation of the cut-tree, it uses the Concorde code. The instances on which we tested the performance are described in the last paragraph of the previous section. The expected improvement can be seen from the running time results shown in Fig. A.4.

11.3 Cactus based heuristic for KCs

In this and the following sections, we give data showing the running times and effectiveness in terms of average numbers of inequalities produced and average time required to find one inequality of the separation algorithms 9.1, 9.2, 9.3, and 9.4. We have implemented these algorithms in C++. The algorithms which rely on a cactus representation of all minimum cuts use K.M. Wenger's cactus implementation [Wen99, Wen03].

First we give results to estimate the effectiveness and efficiency of Algorithm 9.1. Fig A.5 shows the running times of the algorithm on the vertical for the sizes of the support graphs after shrinking, which are in the horizontal direction. Fig A.6 gives the running times in relation to the number of R-sets, also after shrinking. In both figures, for each run of the separation routine, a dot has been plotted with the respective coordinates. Approximately 10000 runs were made; runs in which the cactus did not have a cycle have not been counted in this number or included in the figures. It can be seen that the running times increase only moderately with the number of nodes or number of R-sets.

Figures A.7 and A.8 give, for each instance, the ratio of the number of inequalities per call averaged over the B&C-computation of one instance on the vertical, and the size of the input graph on the horizontal. It can be seen that for the big instances, in average, quite few inequalities are found in each run. However, this is probably caused by a time limit which we imposed on the B&C-solver: In the case of the big instances, the computation does not reach the point where the algorithm works best.

Fig. A.9 shows the average time required to produce one inequality on the vertical, and the size of the input graph on the horizontal, and thus gives the "cost" of an inequality produced by this separation routine. For most instances, the inequalities are relatively cheap, only in some instances the production of inequalities with this algorithm is of questionable value. Runs of the B&C-software in which the number of inequalities found by the algorithm was zero are not included in the figure.

The effect of the inequalities produced with Algorithm 9.1 on the LP lower bounds are dealt with in the next chapter.

11.4 "Path-finder" heuristic for path-bridges

The heuristic in Algorithm 9.2 is the simplest of our separation routines. We give results for two different selection of parameters. The parameter selection A avoids short paths, i.e., paths with only two beads. This means that it prefers to partition the R-sets into longer paths. The parameter settings B punishes short paths only moderately. Figures A.10, A.11, and A.12 give the results for the parameters A, A.13, A.14, and A.15 refer to the parameter selection B. It can be seen that it is a bit less costly to produce the inequalities without avoiding short paths too much. This heuristic is the fastest of all the separation heuristics which we propose.

11.5 Cactus cycles based heuristic for PBs

Algorithm 9.3 was also implemented and its performance tested. Figures A.16, A.17, and A.18 show running times, inequalities per call and time per inequality averaged over each run of the B&C-algorithm. Algorithm 9.3 is the algorithm which produces the smallest number of inequalities. Again, see the next chapter for the gap closures which we were able be obtain using this algorithm.

11.6 Cactus cut-nodes based heuristic for PBs

Algorithm 9.4 is the separation routine which is the most costly in terms of running times, see Fig. A.19. For many instances, it produces a huge number of violated inequalities, see Fig. A.20. However, with increasing size of the instances, this effect is destroyed (see Fig. A.21) by the few calls which are possible within a reasonable time limit. The increase of the time which is required to produce a single inequality is considerable, see Fig. A.22. On the other hand, the algorithm produces inequalities in the cases when the cactus cycles based algorithms fail.

Chapter 12

Solution of the GRP

12.1 Instances

For our computational tests, we have compiled a set of instances of various origins, whose sizes range up to more than 5000 nodes. We describe the groups now.

ALBA Albaida RPP instances described in [CS94]. These instances are all defined on the same graph.

MADR Madrigueras GRP instances, described in [CLS01]. These instances are all defined on the same graph.

GRP See [CLS01]; the underlying graph is that from the ALBA instances.

WPP Instances defined on the basis of Windy Postman Problem instances created by A. Corberán. We created GRP instances from these by selecting one of the two edge costs and a random set of required edges.

dv, stein, vrpfeas A number of instances created by randomly selecting a set of required edges in graphs from various Vehicle Routing and Steiner Tree Problem instances from the OR-Library of J. Beasley.

egl Capacitated Arc Routing instances contributed by R. Eglese. We ignored the demands and capacities to obtain GRP instances.

uni GRP instances created out of electronic circuit board plotting instances contributed by G. Reinelt [Rei94].

alb The graphs for these instances are taken from the Hamiltonian graph testing problems in the TSPLIB [Rei91]. The required edges and costs were selected in a random manner.

h1 Random instances generated by the Type I method described in [HLNH99]. Thanks to Tania Robens for implementing the random instance generator software.

To estimate how well our code is capable of solving the GRP, from the whole set of instances we selected a number of 254 difficult instances of moderate sizes. They are listed in the tables in Appendix B. By difficult we mean that the LP-relaxation of the Ghiani-Laporte polytope consisting of the non-negativity $x \geq \mathbf{0}$, upper bound $x \leq \mathbf{1}$, connectivity (1.1b), and cocircuit (2.3) inequalities, using a *fixed*, arbitrarily chosen Ghiani-Laporte tree does not yield an optimal solution for the instance. We chose instances with $|V(G)|$ not much greater than 2000 to keep the running time and, more importantly, memory requirements moderate.

The data of the instances are displayed in the tables in Appendix C. We note that from the ALBA, MADR, and GRP-sets, *all* instances are listed, for which the relaxation just described did not yield an optimal solution. Some of the instances have name suffixes describing the parameters with which they were created. The name suffix "rq.p" indicates that the probability with which an edge was chosen to be required was $p/10$, and the name part "infty-metr" or "1-metr" refers to the way how the edge costs were computed when the nodes were given as points in the plane: either with the maximum-norm $|\cdot|_\infty$ or with the sum-norm $|\cdot|_1$. For the "alb"-set, the "c0-r0" and "c1-r2" indicate the method with which the required edges and the costs for the edges were chosen.

12.2 A direct comparison to previous work

Corberán, Letchford, and Sanchis [CLS01] gave computational results for a cutting-plane algorithm with a subsequent branch-&-bound phase. They used the ALBA, GRP, MADR, and a number of Hertz et al. [HLNH99] randomly generated instances. We start the description of the performance of our B&C-algorithm by comparing our lower bounds to results of [CLS01], for the ALBA, GRP, and MADR instances (the Hertz et al. instances were not available to us). Tab. 12.1 lists the names of the instances for which our results differ from those in [CLS01]. The second column shows the lower bound which could be obtained by our code. If the instance could be solved by mere cutting plane generation without branching, we just write "opt". The second column shows the lower bound obtained by [CLS01]. The third column gives the value of the optimum solution.

Instance	this thesis root lb	[CLS01] root lb	optimum
ALBA-3-1	5730	5703	5732
ALBA-3-2	opt	6699	6716
ALBA-3-3	6199	6189	6201
ALBA-5-4	opt	4714	4719
GRP04	5091	5119	5186
GRP10	opt	5029	5122
MADR-3-1	opt	8672	8680
MADR-3-3	8552	8526	8555
MADR-3-4	8672	8665	8680
MADR-3-5	opt	8745	8755
MADR-5-3	6948	6949	6955
MADR-5-5	6772	6765	6790

Table 12.1: Comparison on instances of [CLS01]

With our B&C-algorithm, some of the formerly very difficult instances can be solved to optimality without branching, in particular GRP10 and MADR-3-1. We direct the reader to the big gap which the relaxation based on the Ghiani-Laporte polytope with fixed tree leaves for these instances. For the instance GRP04, the lower bound produced by our code was not as good as that in [CLS01].

12.3 Comparison of lower bounds

We start with a theoretical consideration comparing known relaxations of $\mathrm{GL}(\Gamma, T)$ to those of $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$.

12.3.1 Lemma *The LP-relaxations of* $\mathrm{GL}(\Gamma, T)$ *and* $\mathrm{GRP}(\Gamma, \mathbf{b}^T)$ *consisting of non-negativity, upper bound, connectivity, and switched path-bridge (including blossoms* (2.5a)*) inequalities result in the same lower bound. The same is true for the LP-relaxation consisting of non-negativity, upper bound, connectivity, and blossom inequalities.*

Proof. We show that for any $x\colon E(G) \to \mathbb{Q}$ which satisfies all the named inequalities, the vector $x'\colon E(G^T) \to \mathbb{Q}$ defined by $x'_e := x_e$ for all $e \notin T$ and $x'_e := x'_{e^\sim} := \frac{1}{2}x_e$ for all $e \in T$ also satisfies all the named inequalities. To see this, we assume that (a', α') is an inequality of one of the classes which is violated by x', and show that x is also violated by an inequality of one of the classes. By construction of x', this clearly holds for the upper bound constraints, and also for all inequalities (a', α') for which $a'_e = a'_{e^\sim}$ for every $e \in T$. Thus let (a', α') be a switched PB- or cocircuit-inequality satisfying, $a'_e = -a'_{e^\sim}$ for some edges $e \in T$, which in particular means that, w.l.o.g., $a_e = 1$ (by scaling (a, α) accordingly), and if (a, α) is a switched PB-inequality, then $e, e^\sim \in (A : Z)$. For each such edge e, precisely one of $e, e^\sim$ is in the set F, which implies $a'_e x'_e + a'_{e^\sim}(1 - x'_{e^\sim}) = 1$. By an argument similar to the one used to prove Lemma 8.2.1, this implies that (a', α') is not violated. $\square$

Obviously, the statement of the lemma also holds for switched inequalities which flip only R-internal edges. Thus we see that, given the current set of classes of inequalities which we use in our B&C-algorithm, the IP-formulation proposed by Ghiani & Laporte offers no practical advantage, but increases the number of variables.

12.3.1 Comparison of lower bounds for a mere cutting-plane approach

For the remainder of this section, we will give computational results for various relaxations obtained by the separation routines which we introduced in Chapter 9. We use the late selection of the tree T, see 10.4.2, and the strategy described in 10.4.1. We investigate to what extent the *Ghiani-Laporte gap* can be closed in this way. By Ghiani-Laporte gap, we mean the gap of the *Ghiani-Laporte lower bound,* which is obtained by optimizing, for a *fixed* T, over the LP-relaxation given by the connectivity (1.1b), non-negativity (1.1c), and blossom inequalities (2.5a). For some very difficult instances, the value of an optimal solution is not known, so the gaps are defined in terms of the best known upper bounds.

Ghiani-Laporte lower bound

We use the Branch-and-Cut software framework **ABACUS** [JT98] for all generic B&C-mechanisms such as storing the tree of subproblems, selecting a subproblem, interfacing the LP-solver (Cplex).

We have chosen the following way to visualize the data. The 254 instances are on the horizontal of 2-D plots. For each piece of data we want to visualize, we plot a point above the respective instance, in the height determined by the piece of data. The instances are arranged in a sorted way, so that the sizes of the instances increase from left to right. Fig. C.1 graphically displays the sizes of the 254 instances: the upper line designates the number of nodes $n := |V(G)|$, the lower line gives the number of R-sets $|\mathcal{C}|$. All graphs are sparse (see the tables Appendix C).

The Ghiani-Laporte gap of the set of instances is graphically visualized in Figures C.2 and C.3. The vertical axis shows the gap in per cent between the lower bound and the best known upper bound, namely the value

$$100\frac{\text{ub} - \text{lb}}{\text{lb}}.$$

where ub denotes the value of the best known upper bound, and lb is the value of the lower bound.

Figures C.4 to C.8 show the gap closure

$$\frac{\text{ub} - \text{lb}}{\text{ub} - \text{gllb}},$$

where gllb denotes the Ghiani-Laporte lower bound of the instance, and lb the lower bound which could be achieved within a time limit of 3000 seconds. The case gllb = ub is visualized by plotting the dot in the negative half of the vertical axis (value -5). Values exceeding 109 are plotted with vertical coordinate 109.

Figures C.9 to C.13 shows the relative change of the lower bound with respect to the Ghiani-Laporte lower bound in per cent, i.e., the value $100^{\text{lb}}/_{\text{gllb}}$. Only the interesting region between -1 and 5 per cent is shown.

A note on the lower bounds is in place. The time limit imposed on the optimization might be reached or the code may crash because of memory exhaustion before the separation of blossom inequalities catches on. In this way, having not added important blossom inequalities to the relaxation might result in a deterioration of the gap.

For difficult instances, the code occasionally crashes for one of the following two reasons: (a) memory exhaustion—this is caused mainly by the huge numbers of path inequalities which are found; (b) the cactus construction code [Wen03] fails to produce a cactus due to floating point precision problems (this is not a bug in the cactus code)—when that happens, the cactus construction code aborts the entire program execution. These crashes (and sometimes the time limit) are the reason why, for very difficult instances, the lower bounds are sometimes quite bad, worse in fact than the Ghiani-Laporte lower bound.

Figures C.14 to C.18 give the running times in which the bound was achieved and Figures C.19 to C.23 give the number of LPs which had to be solved in the process. In these figures, a code crash is indicated by a negative vertical value of a dot.

All separation heuristics which we have discussed produce a huge number of violated inequalities. The question which we now address is, which heuristics produce the more import cuts.

The KC-algorithm 9.1 turns out to be very successful in increasing the lower bound. Figures C.4, C.9, C.14, and C.19, show the computational results for the attempt to produce the best possible lower bound using only this separation routine on top of the Ghiani-Laporte bound. Despite its simplicity, the KC-heuristic appears to produce very good cutting planes. In may cases, using it alone and omitting the other separation heuristics produces as good upper bound improvements as using it together with the PB-separation routines. These good lower bounds can even be achieved in relatively short time, and only a moderate number of LPs need to be solved.

The "path-finder" heuristic, Algorithm 9.2, and the PB-heuristic based on the cactus, Algorithm 9.3, are the ones which produce the weakest cuts, when compared to the other heuristics discussed in Chapter 9. This can be seen in the Figures C.5, C.10, C.15, and C.20, for the path-finder heuristic, and Figures C.6, C.11, C.16, and C.21 for the cactus based PB-heuristic. We would like to point out that even though these heuristics are less effective as the others, they allow to reduce the gap of many of the instances to zero. Even for the instances with 1000 nodes and more, the gap which remains between the Ghiani-Laporte lower bound and the optimum can be significantly reduced. Also remember that these two heuristics are considerably faster (per call) than the other two (as discussed in the previous chapter).

Figures C.7, C.12, C.17, and C.22 display the results for the cactus cut-nodes based PB-heuristic 9.4. On its own, it does not produce enough of the important KC-inequalities which are required to increase the lower bound. However, it turns out to be very useful for many of the instances when used together with the KC-heuristic, because it can detect path-bridge configurations in support graphs without having to rely on the presence of cycles in the cactus.

Figures C.8 and C.13 show how much the lower bounds can be increased using simultaneously all the separation routines and techniques which we have discussed, and also the KC- and HC-separation routines described in [CLS01]. There is only a small average improvement over the results of the KC-heuristic, but there are a number of instances for which the lower bounds increase significantly or which can even be solved to optimality in the root node. This may in part be caused by the usage of the HC-inequalities, and by the fact that the PB-inequalities show their strength when used on top of the KC-inequalities.

12.4 Solution by Branch-and-Cut

Up to now we have only considered cutting-plane generation. Now we present computational results for the full B&C-algorithm. Figures C.24 to C.30 refer to a single run of our B&C-code with a fixed setting of parameters for all 254 instances. Again, a time limit of 3000 seconds was imposed. In contrast to the five runs of our code as cutting-plane algorithm, we used the upper bound heuristic 10.4. Fig. C.24 shows the gap closure relative to the Ghiani-Laporte gap. It can be seen that the majority of instances can be solved to optimality within the time limit. For some instances, the gap can be reduced significantly. On a number of instances, no improvement was possible, or the Ghiani-Laporte lower bound could not even be reached (see the comments in Section 12.3 above). Fig. C.25 gives the change of the lower bound in per cent relative to the Ghiani-Laporte lower bound.

Figures C.26 to C.29 show the running times in seconds of the B&C-code. If the code crashed on the instance, a negative value is plotted, and the points in the line near the top frame indicate that the computation took longer than the time limit given in the caption. All except 7 of the 110 instances on graphs with up to 400 nodes could be solved within 100 seconds. Recall that the biggest instances which have been considered for the solution by B&C by [GL00] had at most 350 nodes.

In Fig. C.30, the number of subproblems which were generated in the B&C-process are displayed. Except for two instances, uni3.1.1-metr_4 and uni3.1.infty-metr_4, which required 35 and 37 subproblems to before they could be solved to optimality (not within the time limit, though), the number of subproblems hardly exceeds 20 for any of the instances. This is a result of the strategies described in Chapter 10, namely the attempt to produce many and good cuts early in the cutting plane process, the storing and checking of the paths which were found, and the addition of some of the upper bounds for the edges when no other inequalities were found. Fig. C.31 shows the number of LPs which were computed in the B&C-process. In both figures, a crash of the code is again indicated by a negative value.

Out of the 254 instances, 55 could not be solved to optimality in our B&C-code run within 3000 seconds. For another 12 instances, the optimal lower bound was found, but no upper bound with the same value. The smallest instance which could not be solved, uni2.1.1-metr_6, has 477 nodes and is a GTSP instance. The second smallest instance, steinc7.rq.066, has 500 nodes and 430 R-sets.

We have compiled a list of instances which remain challenging, see Table C.1. This table contains the 61 instances which could not be solved to optimality by our B&C-code run within 1500 seconds, either because the solution took longer or because the code crashed, see also Fig. C.26. The first column indicates the index of the instance in the figures, i.e., the horizontal position in the figures. The columns "best lb" and "best ub" indicate the best lower and upper bounds respectively for the instance which we know. A star "*" in the "best lb" column indicates that upper and lower bound are optimal. The word "opt" in the "best ub" column indicates that the B&C-code could find the optimal lower bound, possibly with the help of branching. In the last column, the running time of the B&C-code is indicated if it was below the 3000 seconds limit. The entry "time limit" indicates that this limit was exceeded, and "mem/flt" indicates that the code crashed due to memory exhaustion or floating point precision problems in the cactus construction.

12.4.1 Conclusion of the computational part

There are 31 instances out of the 254, for which we could not find an optimal solution. For all others, both the optimal lower bound and the optimal upper bound were found in one of the six runs which we have described in the previous chapter, though possibly not both upper and lower bound in the same run. Of the 54 instances with 1000 nodes or more, only 21 remain, for which we could not find and prove an optimal solution with the code runs described in the previous chapter. The region of sizes of challenging instances moved from between 200 ([CLS01]) and 350 ([GL00]) to above 850, and we could solve many instances in the magnitude of 1000 nodes and more.

These results show that there is a considerable potential in the inequalities, separation methods, and other strategies we propose in this thesis. Challenges remain above all in the area of software engineering. A number of promising ways to improve the running times and robustness of our code can be named.

1. The memory management should be improved, in particular the way in which inequalities are stored. The storage of the path lists could be improved. Applegate et al. [ABCC03] propose a way to store inequalities for the STSP. It might be worth while to implement their ideas for the GRP.

2. When the cactus construction fails due to floating point precision problems, the B&C-algorithm should be able to continue. Klaus Wenger's cactus construction code should

be changed so that it does not abort the program execution in this case.

3. The order in which the separation routines are invoked could be optimized to find the order which is best for the majority of instances in the testbed.

4. Parameters could be fine tuned with the aim of improving running time of the separation routines. In particular of the very slow Algorithm 9.4 and the checking of path inequalities could be greatly improved this way.

5. An item which is closely related to memory usage is the possibility to select only a subset of the huge amount of inequalities produced by some separation algorithms for many of the instances, while the other inequalities could be immediately removed from the memory. To do this, the difficult question of deciding which violated inequalities of the same kind are more important than others must be attacked. It is *comparatively* well understood which *kinds* of inequalities are to prefer, e.g., PBs and KCs with long paths and many edges on the paths. However, we guess that the question which inequalities of the same kind are more important is much more difficult.

6. In 10.4.2, we propose a strategy for the late selection of the Ghiani-Laporte tree. While the idea of delaying the choice of edges which receive upper bound one is promising, different heuristic strategies to perform the selection could be invented and compared. Also, currently, the running time of our selection method is quite high, and it could possibly be reduced by fine tuning parameters.

Appendix A

Figures for Chapter 11

A.1 The core minimum odd-cut algorithm and the blossom-minimization core

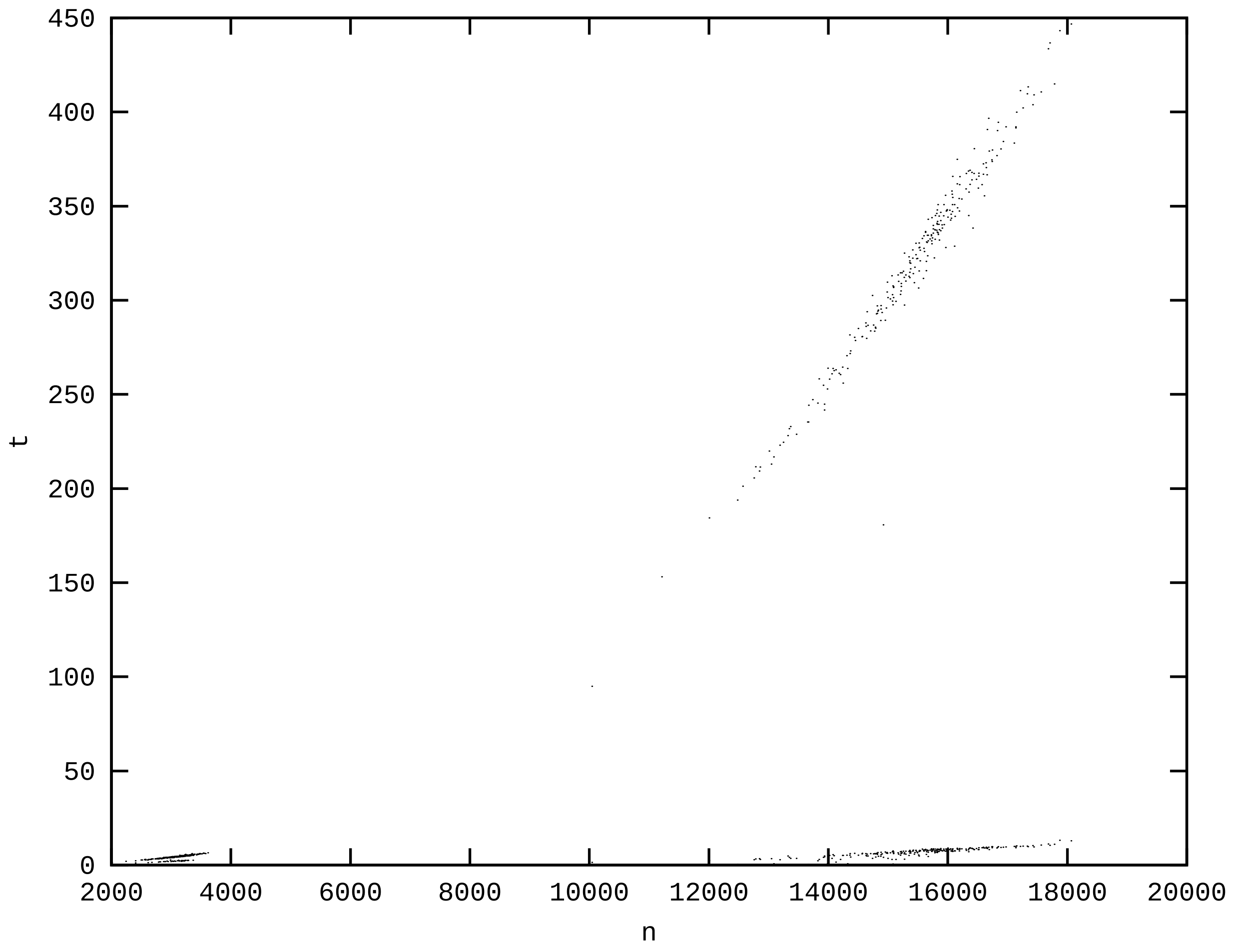


Figure A.1: Running times of HeMOC and Concorde/Gomory-Hu, average degree 3.6

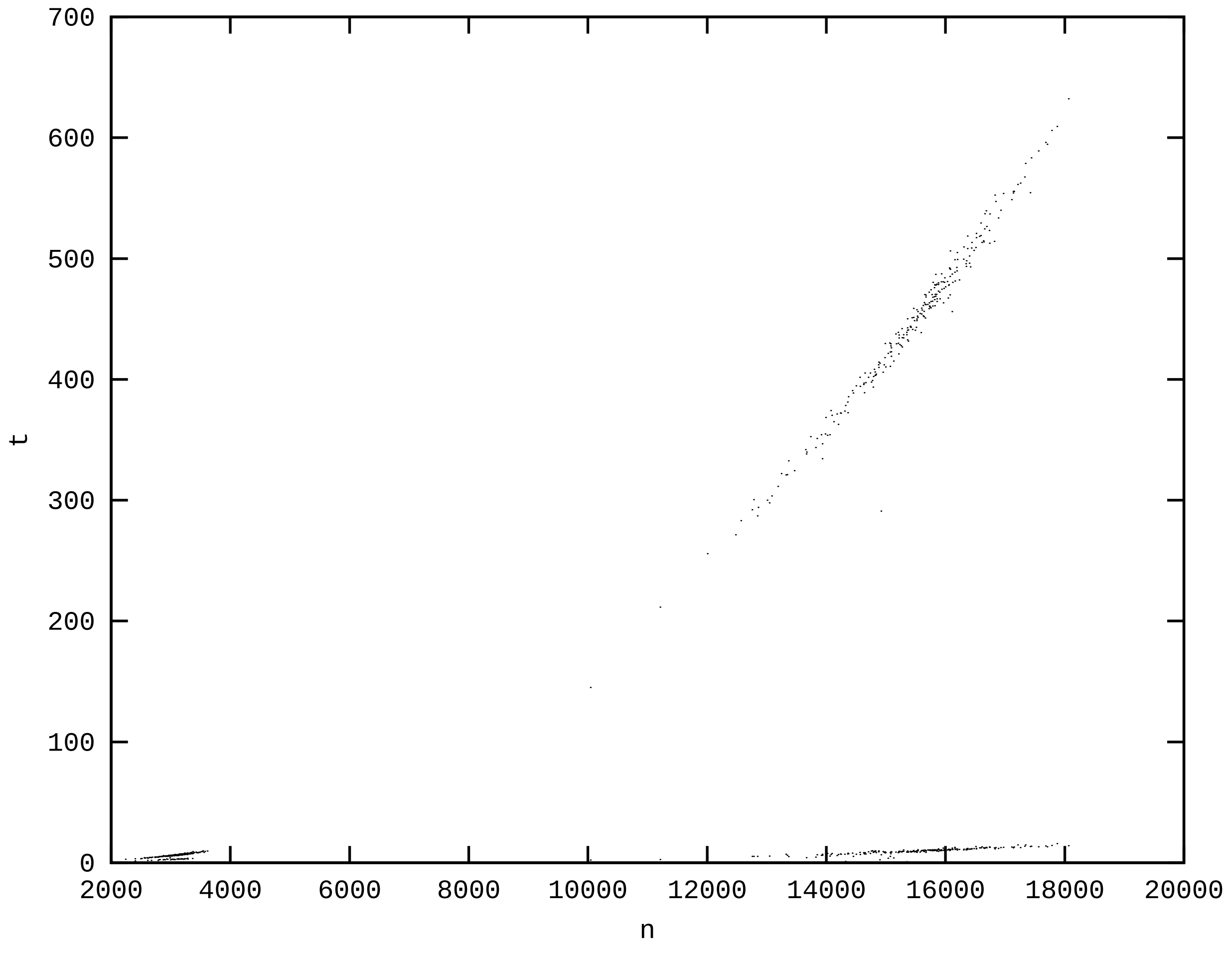


Figure A.2: Running times of HeMOC and Concorde/Gomory-Hu, average degree 4.4

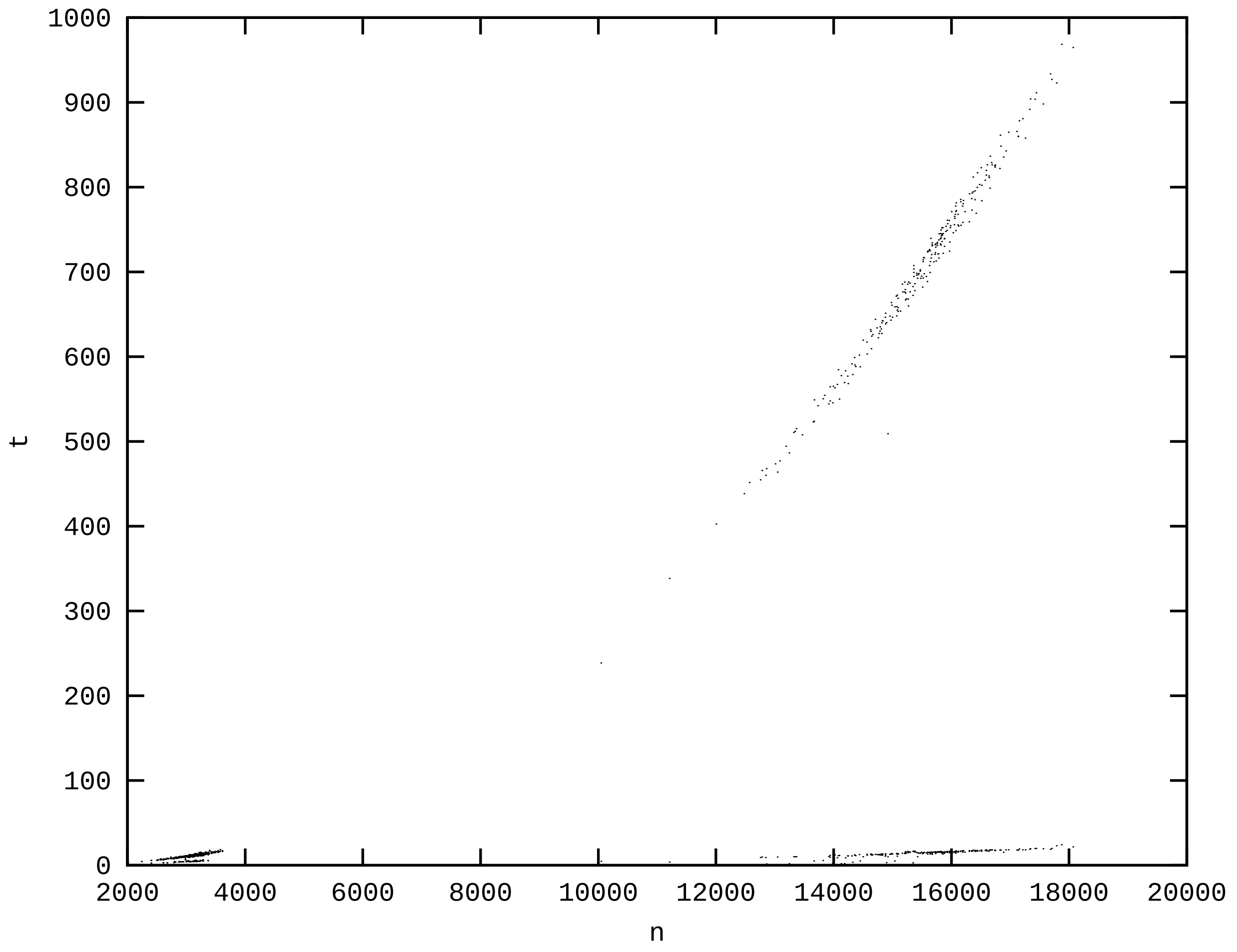


Figure A.3: Running times of HeMOC and Concorde/Gomory-Hu, average degree 6.4

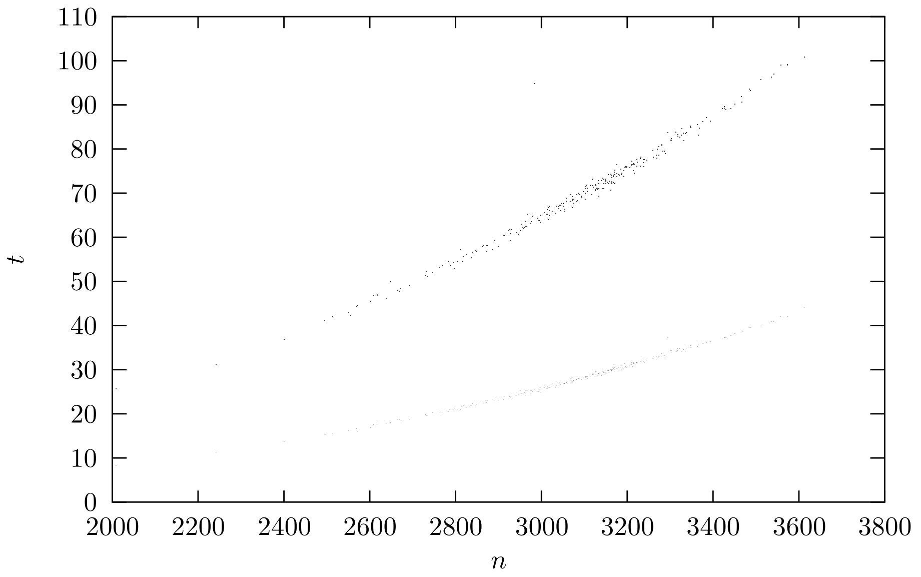


Figure A.4: Running times of Algorithm 7.5 and Concorde

A.2 Cactus based heuristic for KCs

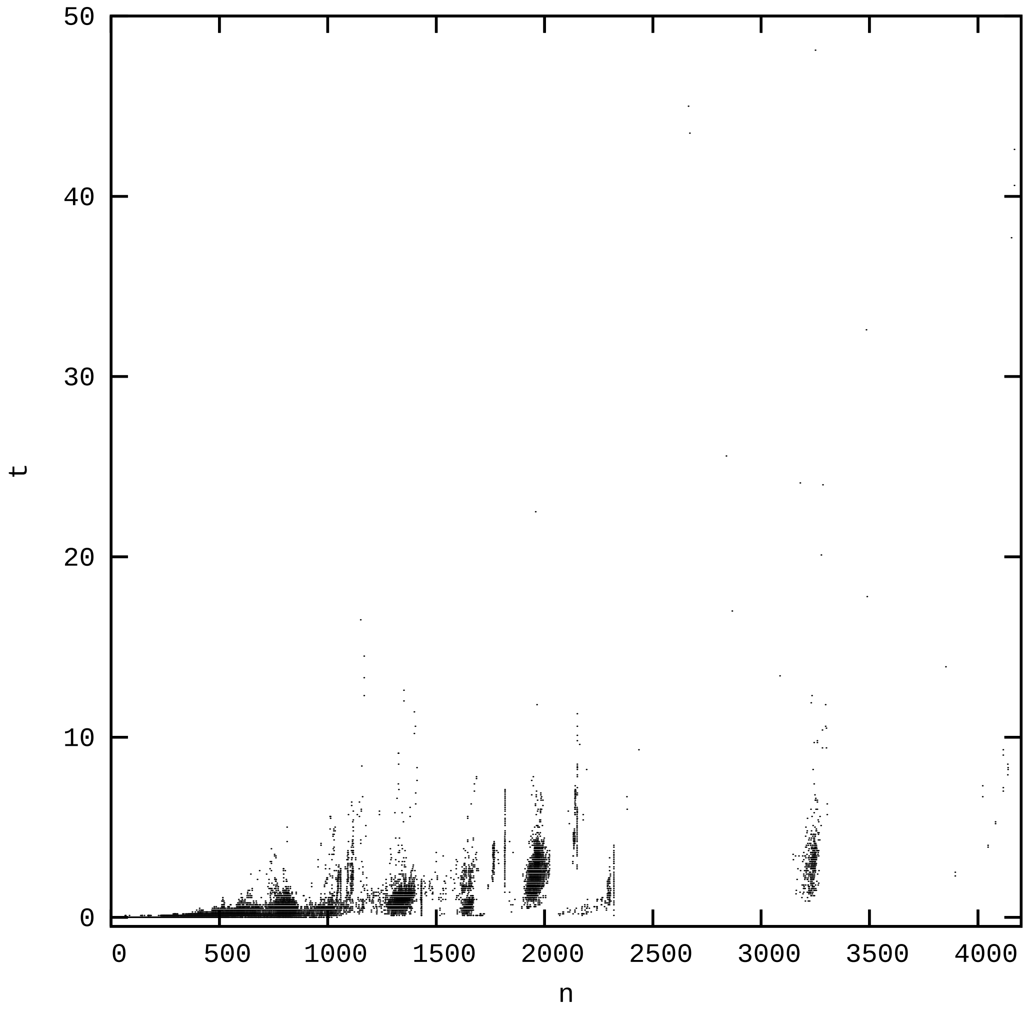


Figure A.5: Cactus based KC-heuristic: running time, $n := |V(G(x^*))|$

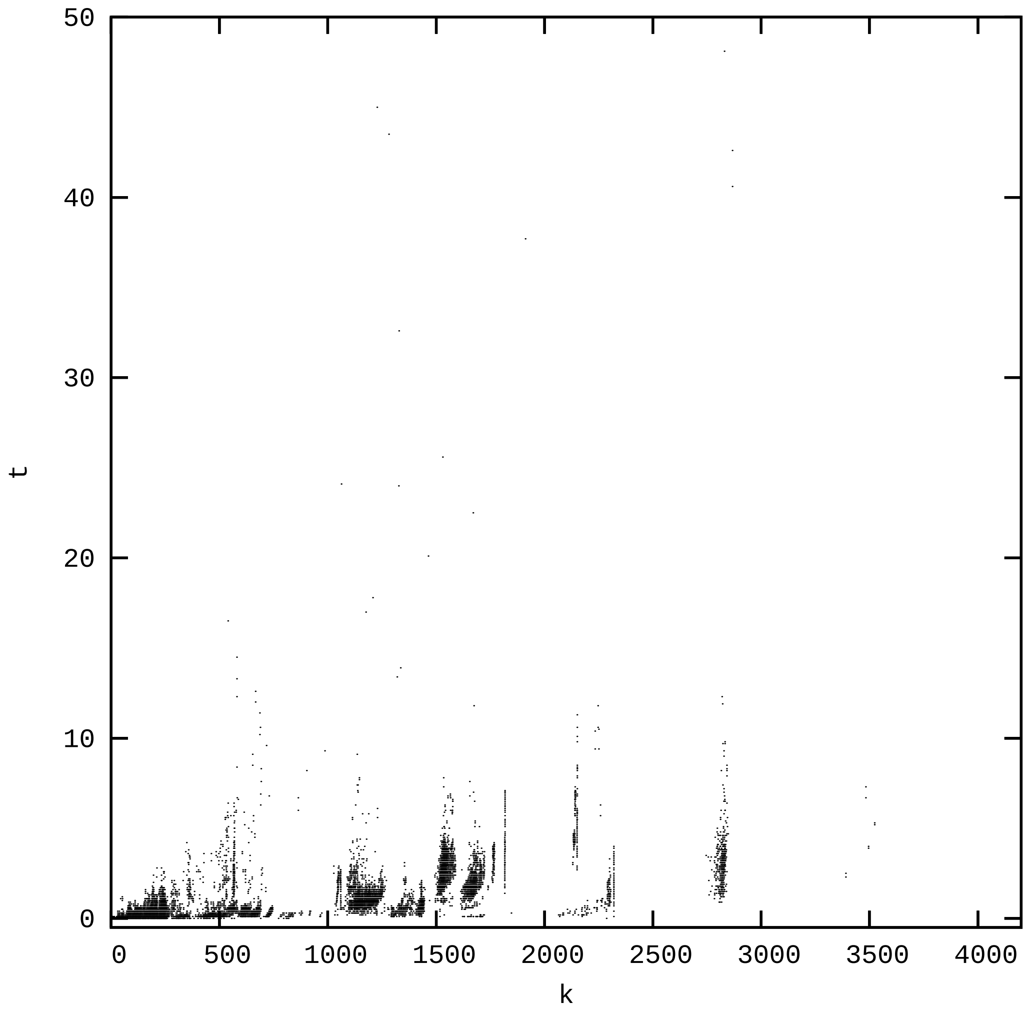


Figure A.6: Cactus based KC-heuristic: running time, $k := |\mathcal{C}|$

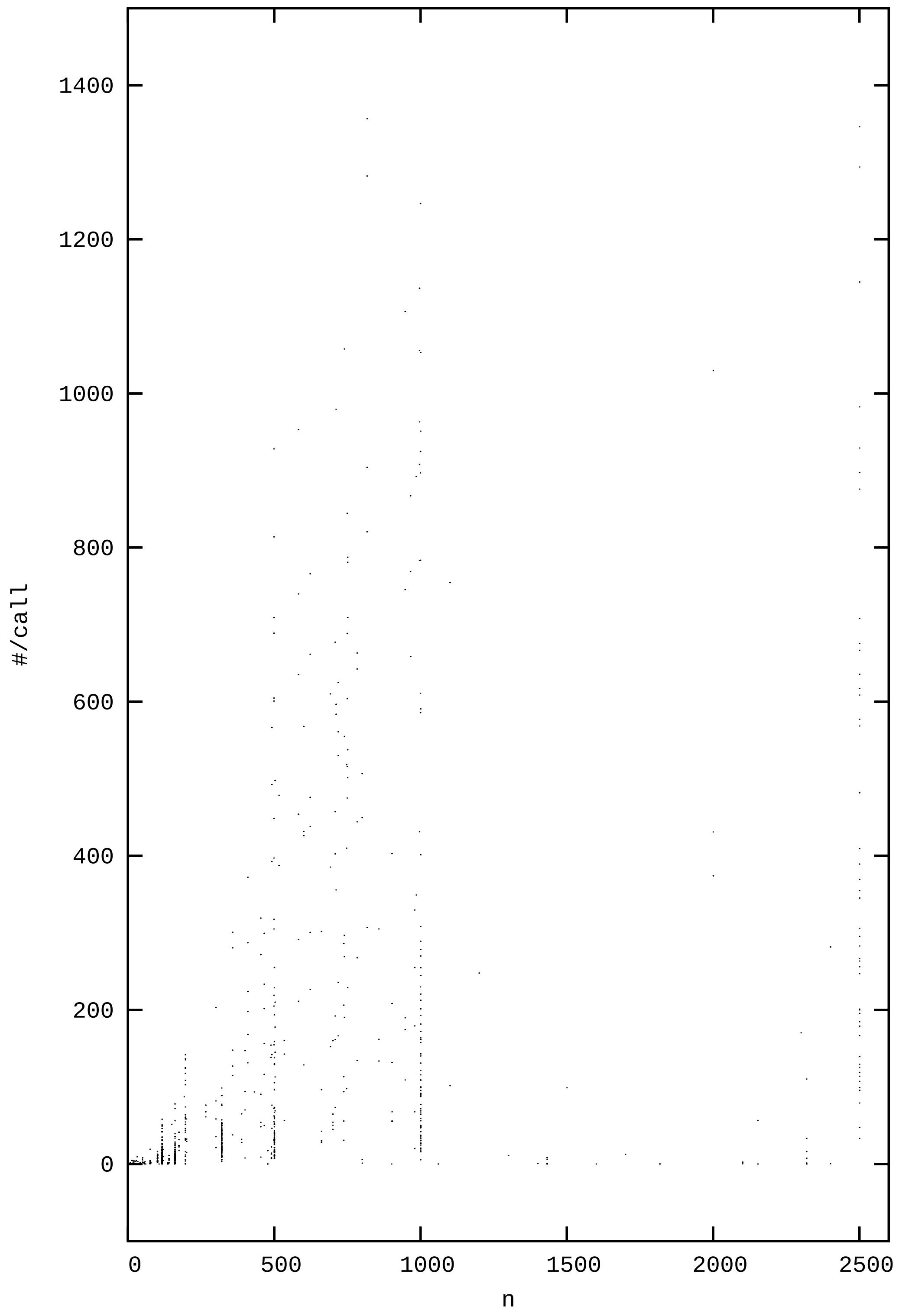


Figure A.7: Cactus based KC-heuristic: inequalities per call, $n := |V(G)|$

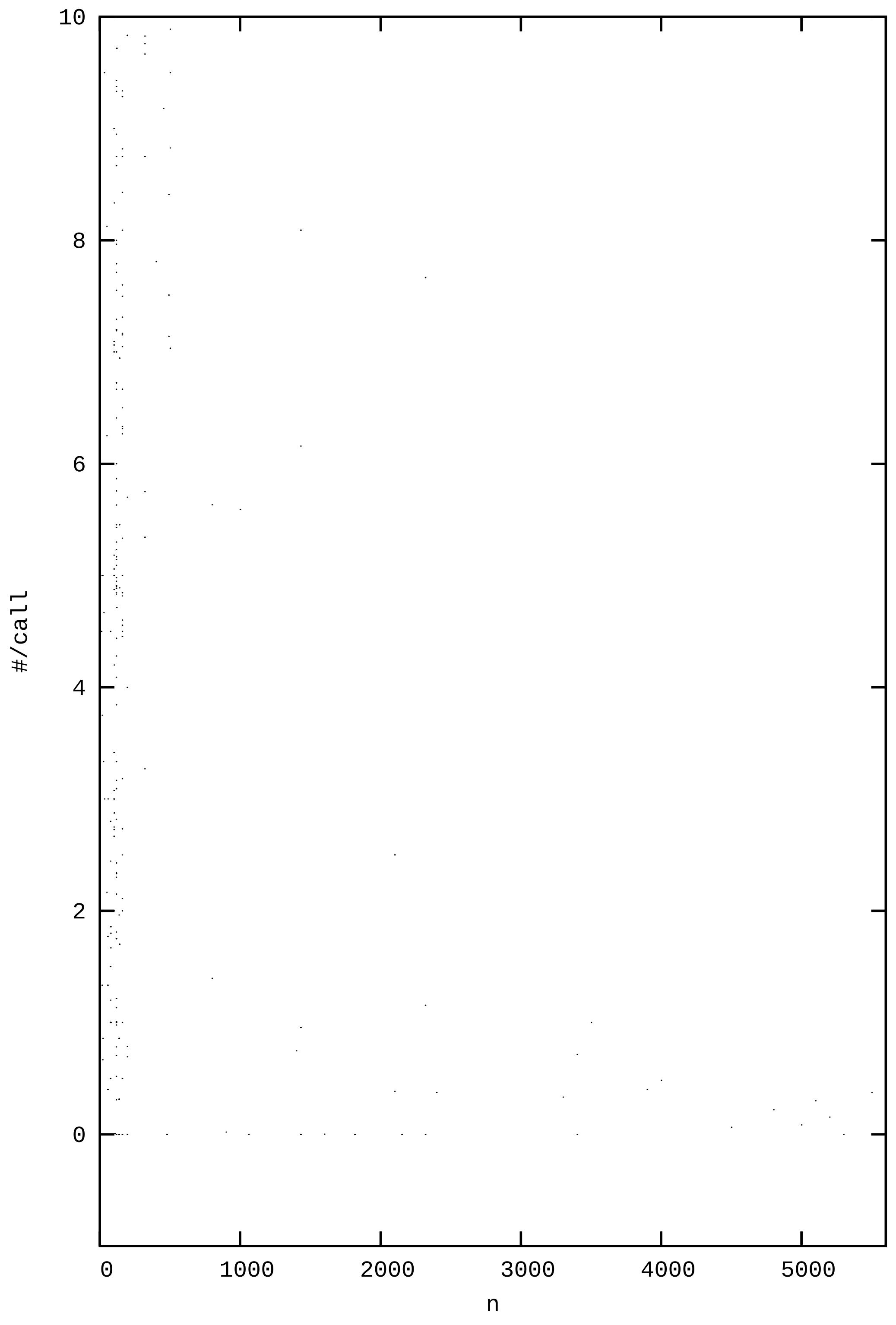


Figure A.8: Cactus based KC-heuristic: inequalities per call, zoomed, $n := |V(G)|$

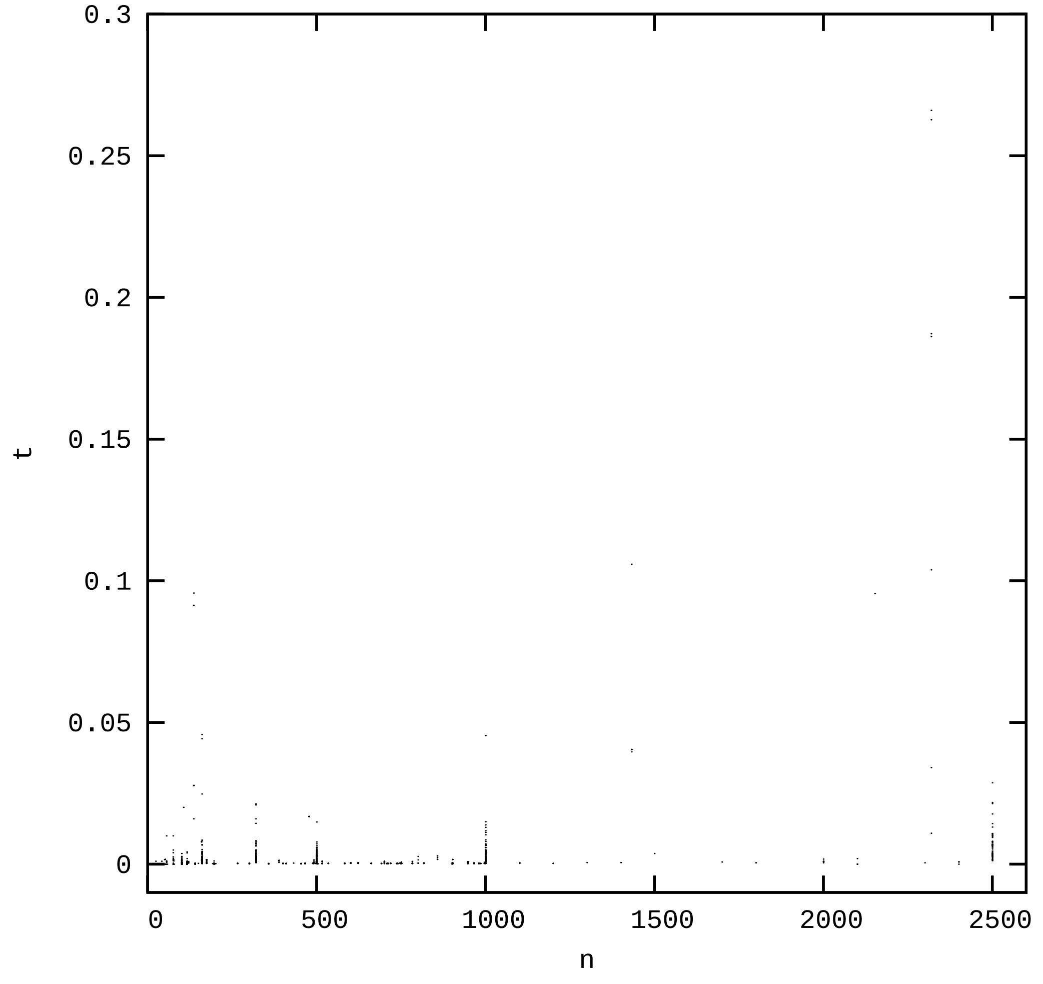


Figure A.9: Cactus based KC-heuristic: time per inequality, $n := |V(G)|$

A.3 "Path-finder" heuristic for path-bridges

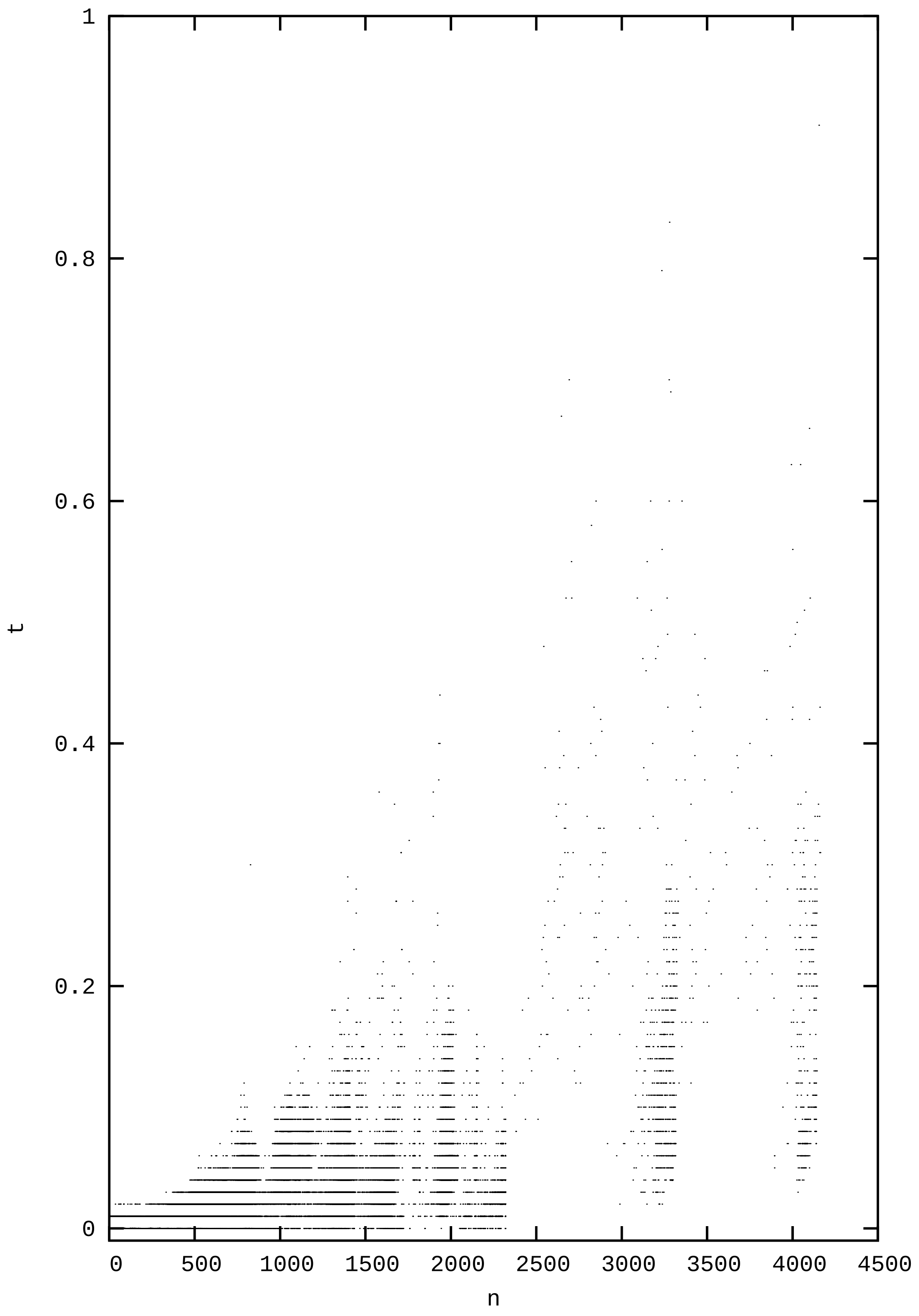


Figure A.10: "Path-finder" PB-heuristic, parameters A: running time, $n := |V(G(x^*))|$

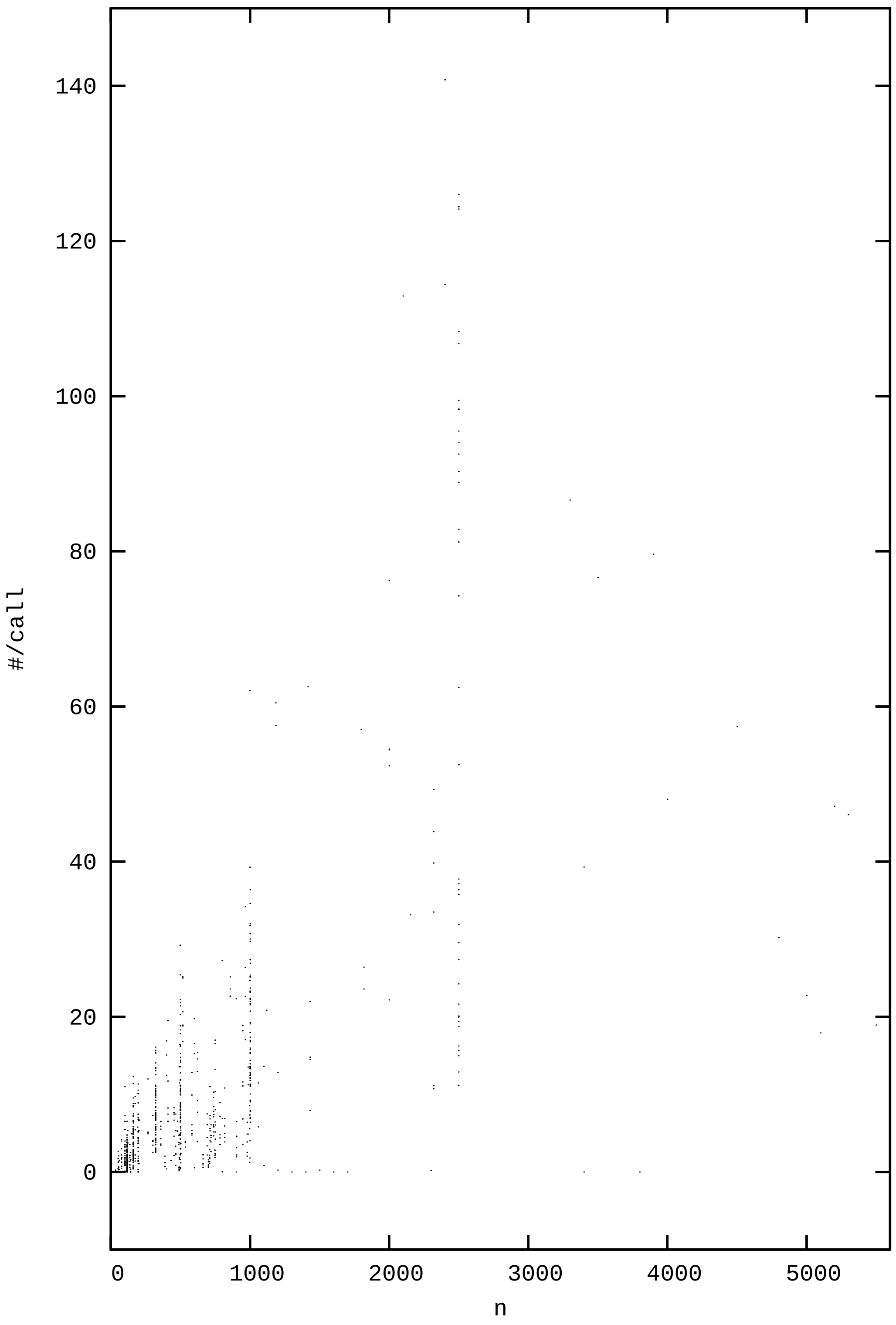


Figure A.11: “Path-finder” PB-heuristic, parameters A: inequalities per call, $n := |V(G)|$

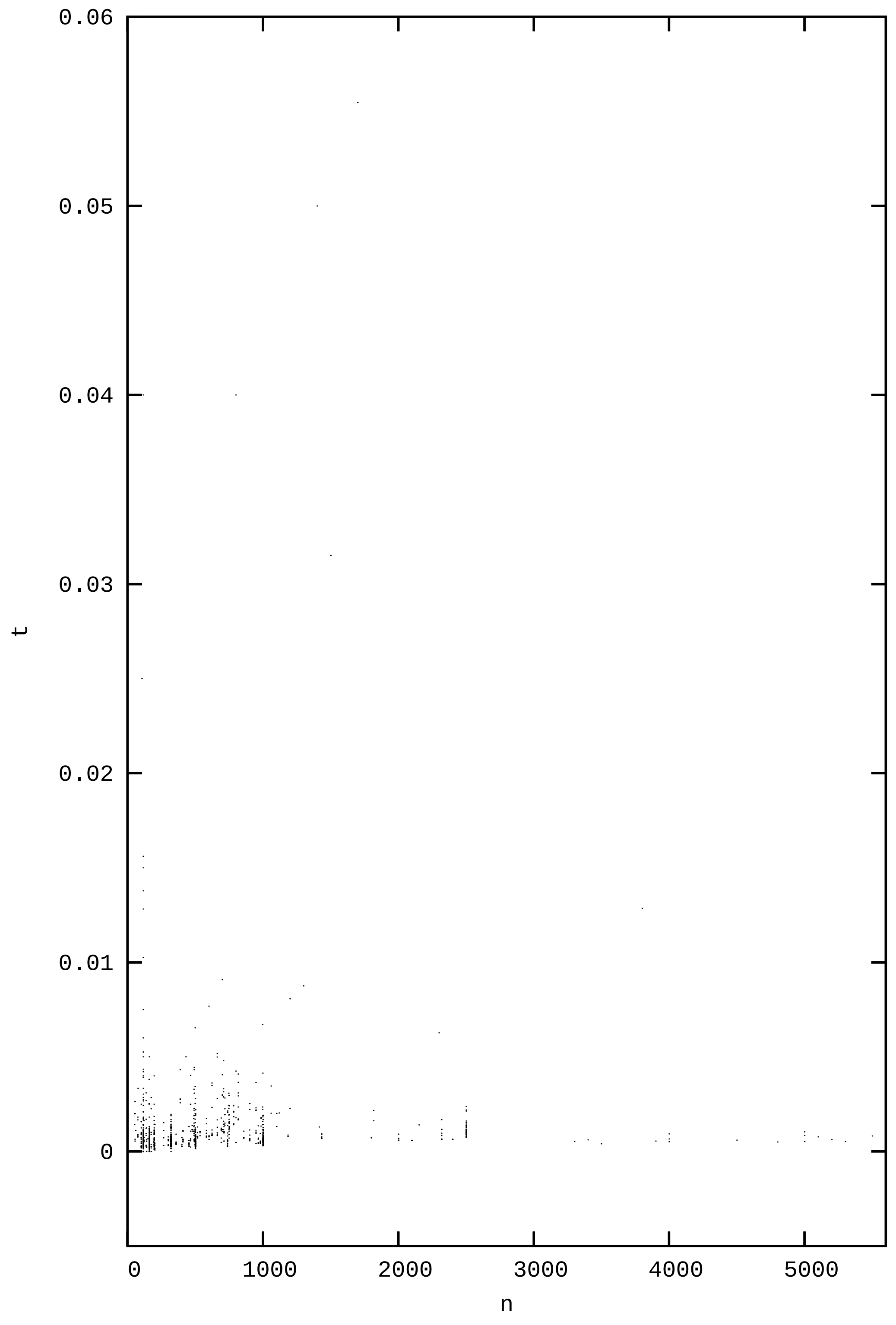


Figure A.12: "Path-finder" PB-heuristic, parameters A: time per inequality, $n := |V(G)|$

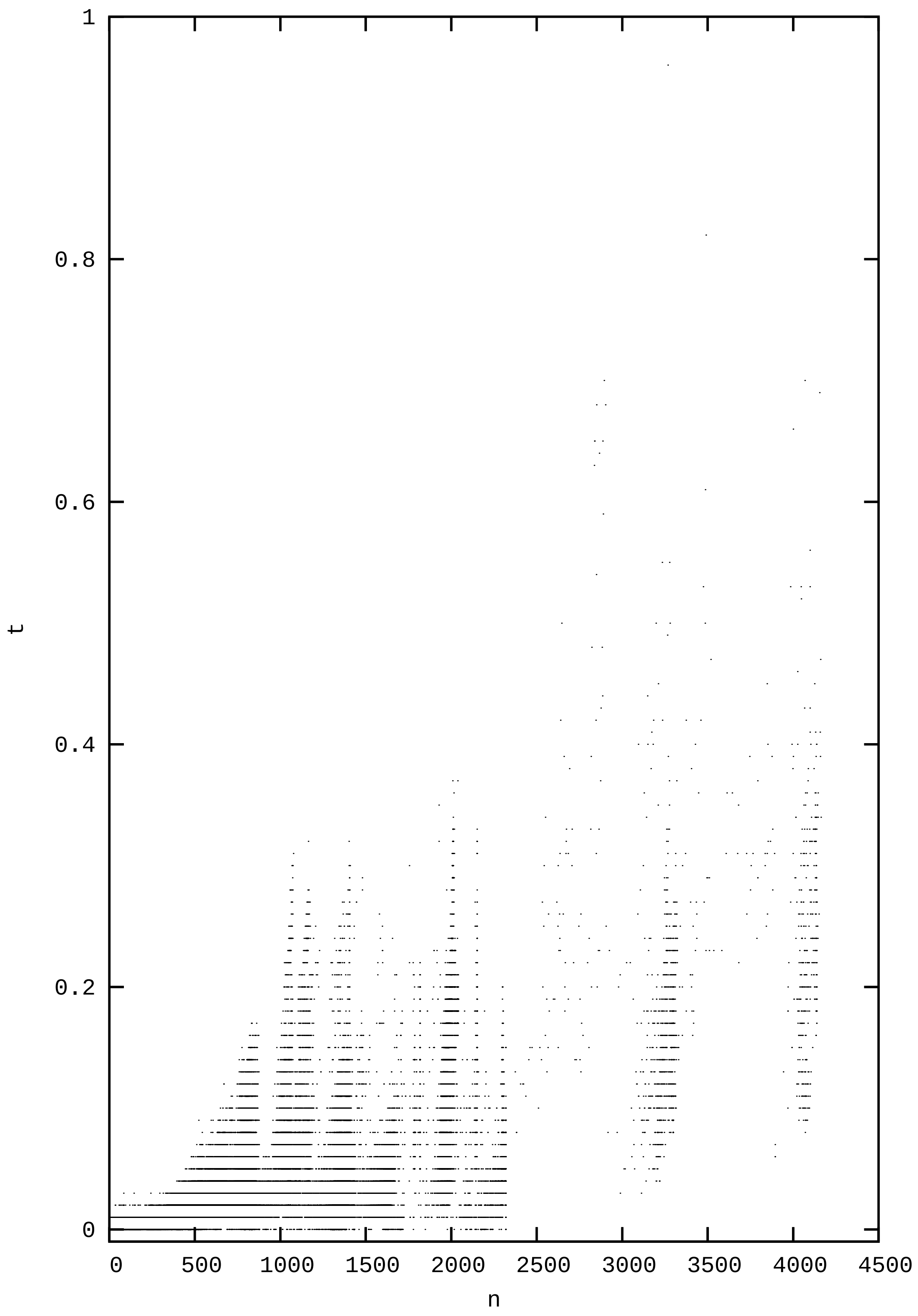


Figure A.13: "Path-finder" PB-heuristic, parameters B: running time, $n := |V(G(x^*))|$

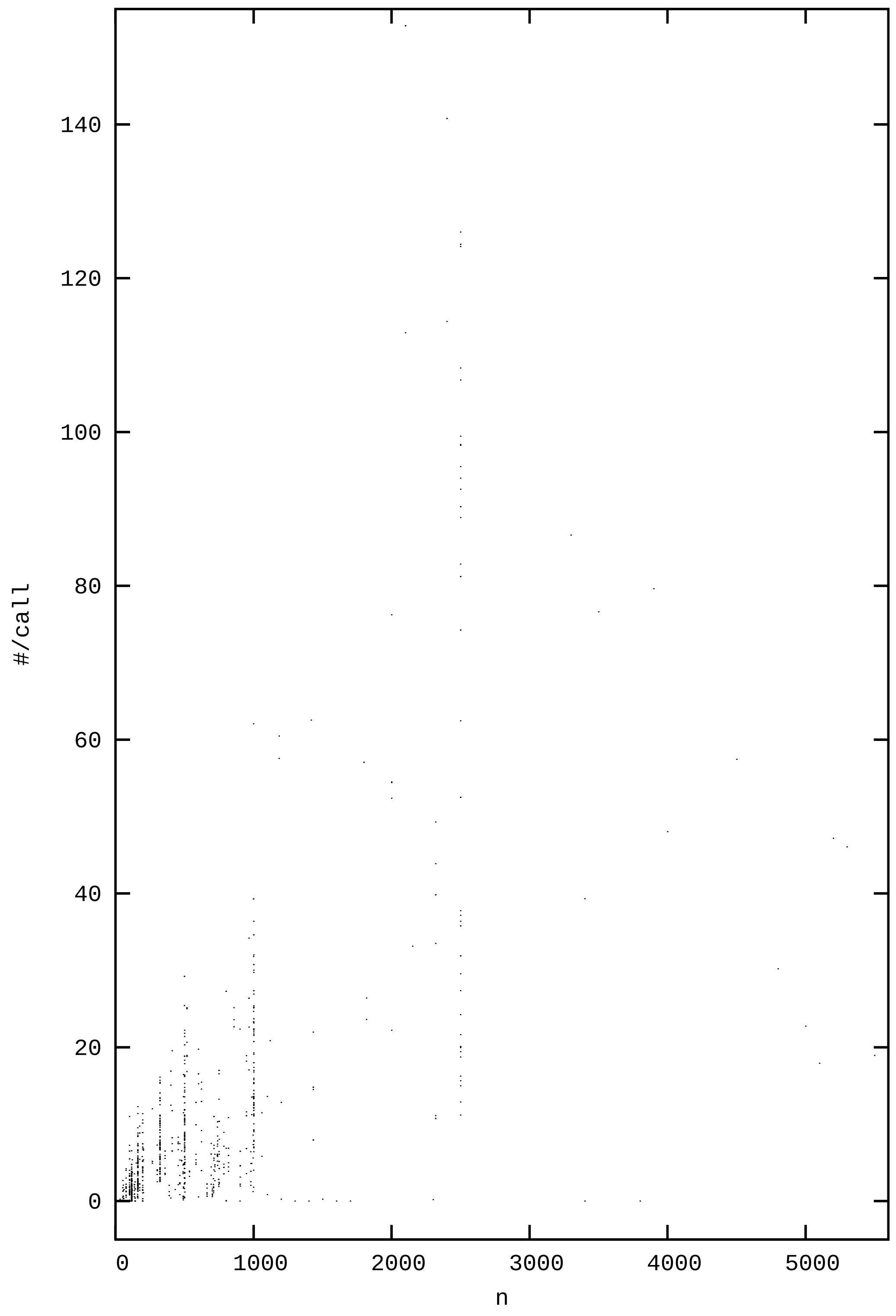


Figure A.14: "Path-finder" PB-heuristic, parameters B: inequalities per call, $n := |V(G)|$

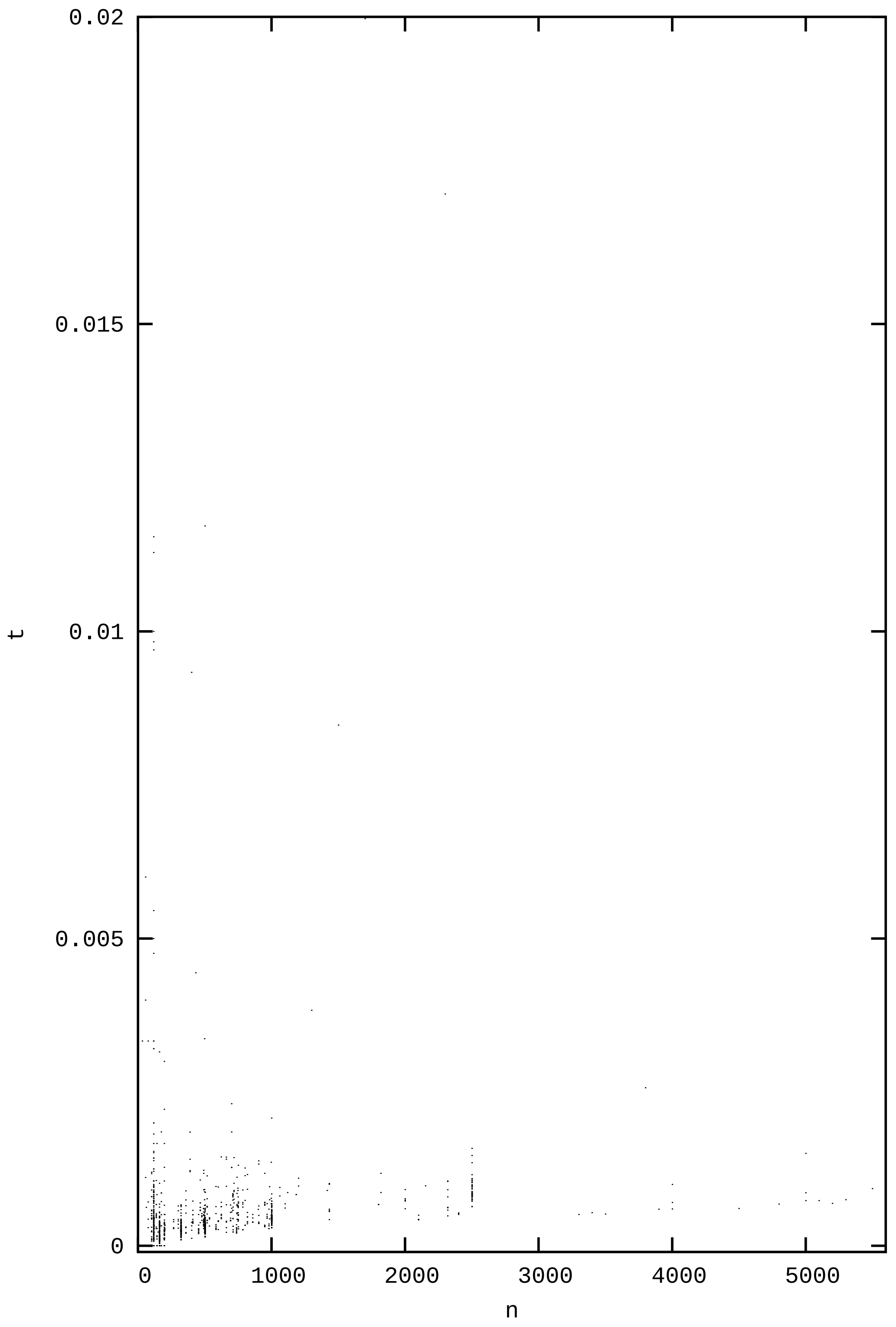


Figure A.15: “Path-finder” PB-heuristic, parameters B: time per inequality, $n := |V(G)|$

A.4 Cactus cycles based heuristic for PBs

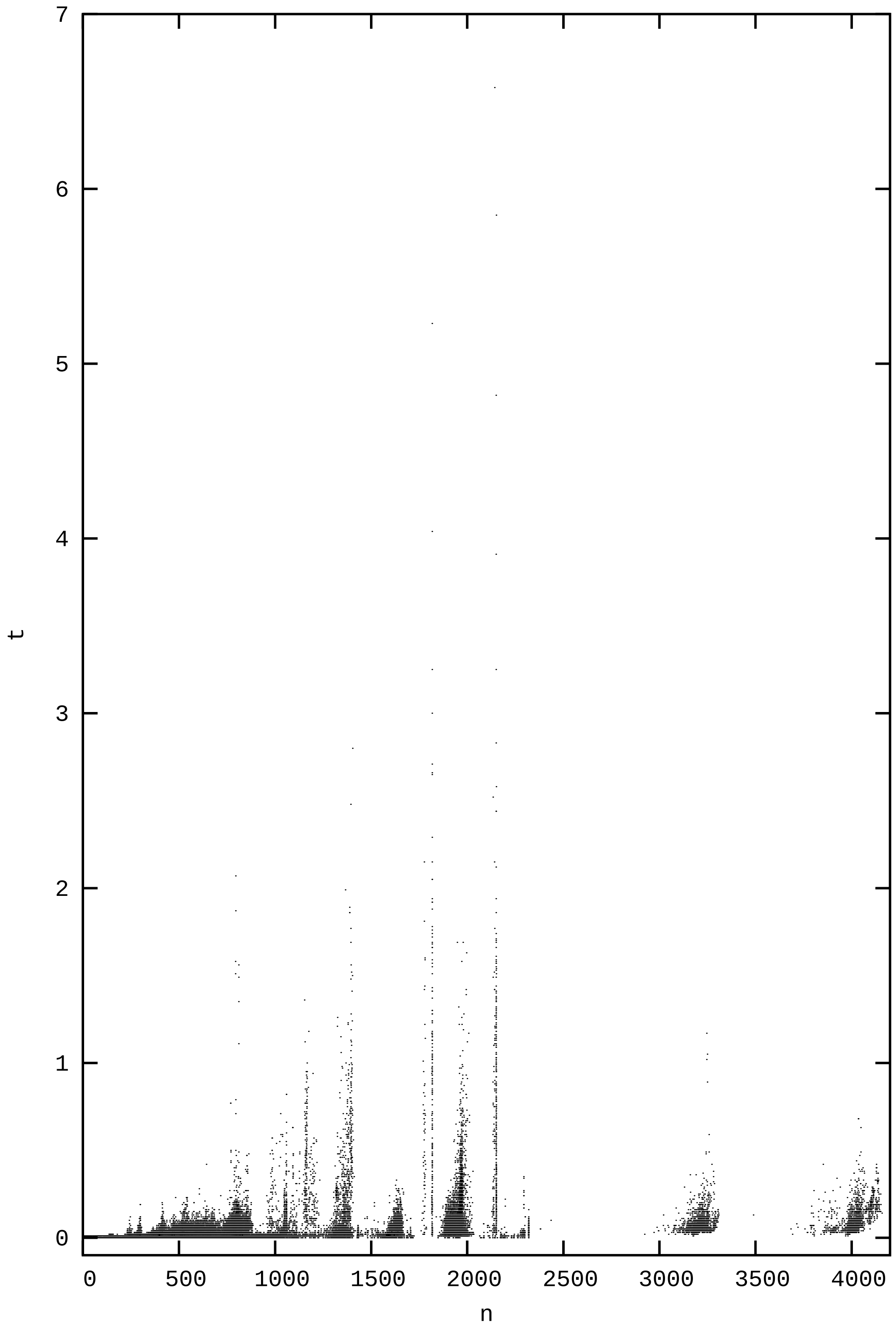


Figure A.16: Cactus cycles based PB-heuristic: running time, $n := |V(G(x^*))|$

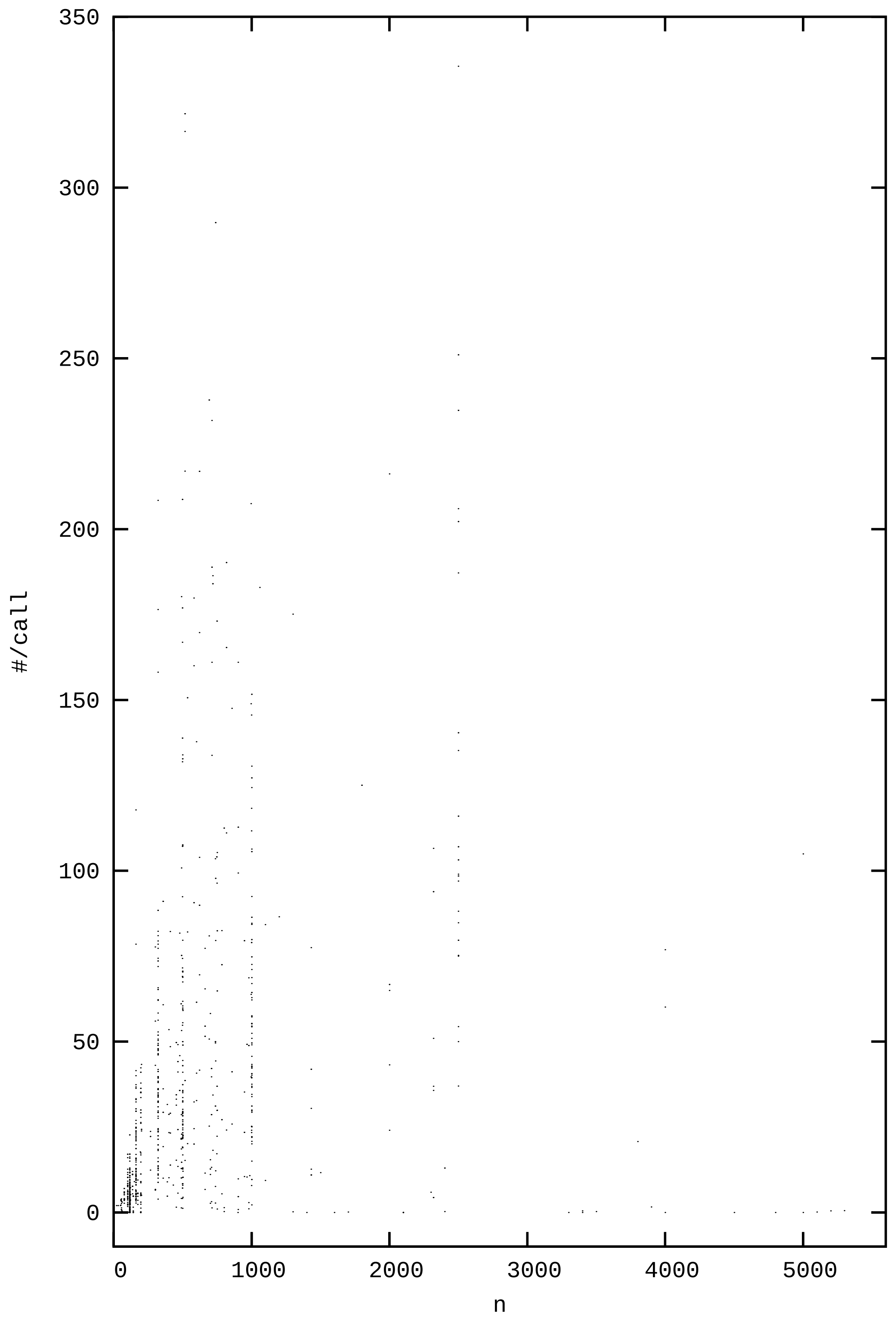


Figure A.17: Cactus cycles based PB-heuristic: inequalities per call, $n := |V(G)|$

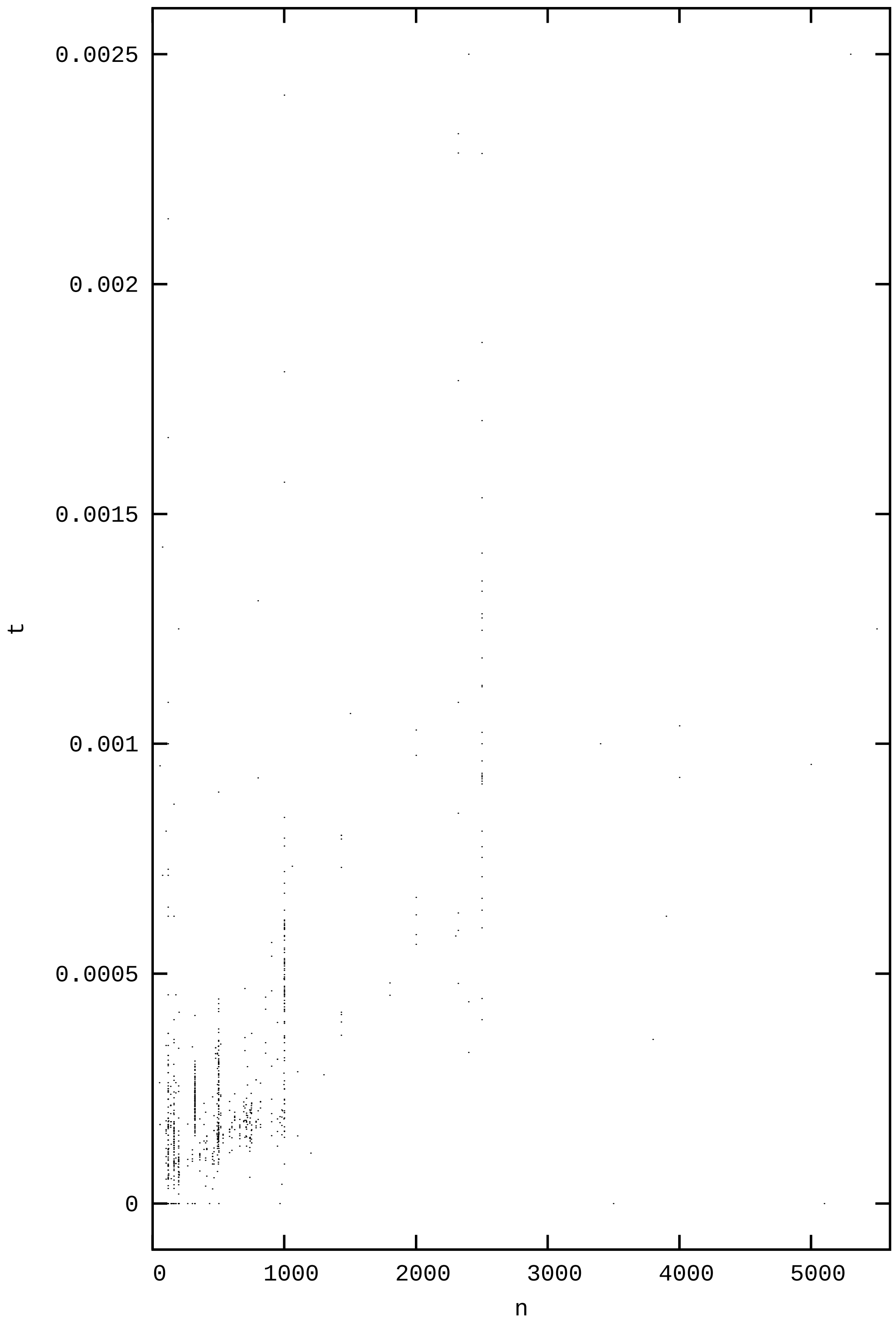


Figure A.18: Cactus cycles based PB-heuristic: time per inequality, $n := |V(G)|$

A.5 Cactus cut-nodes based heuristic for PBs

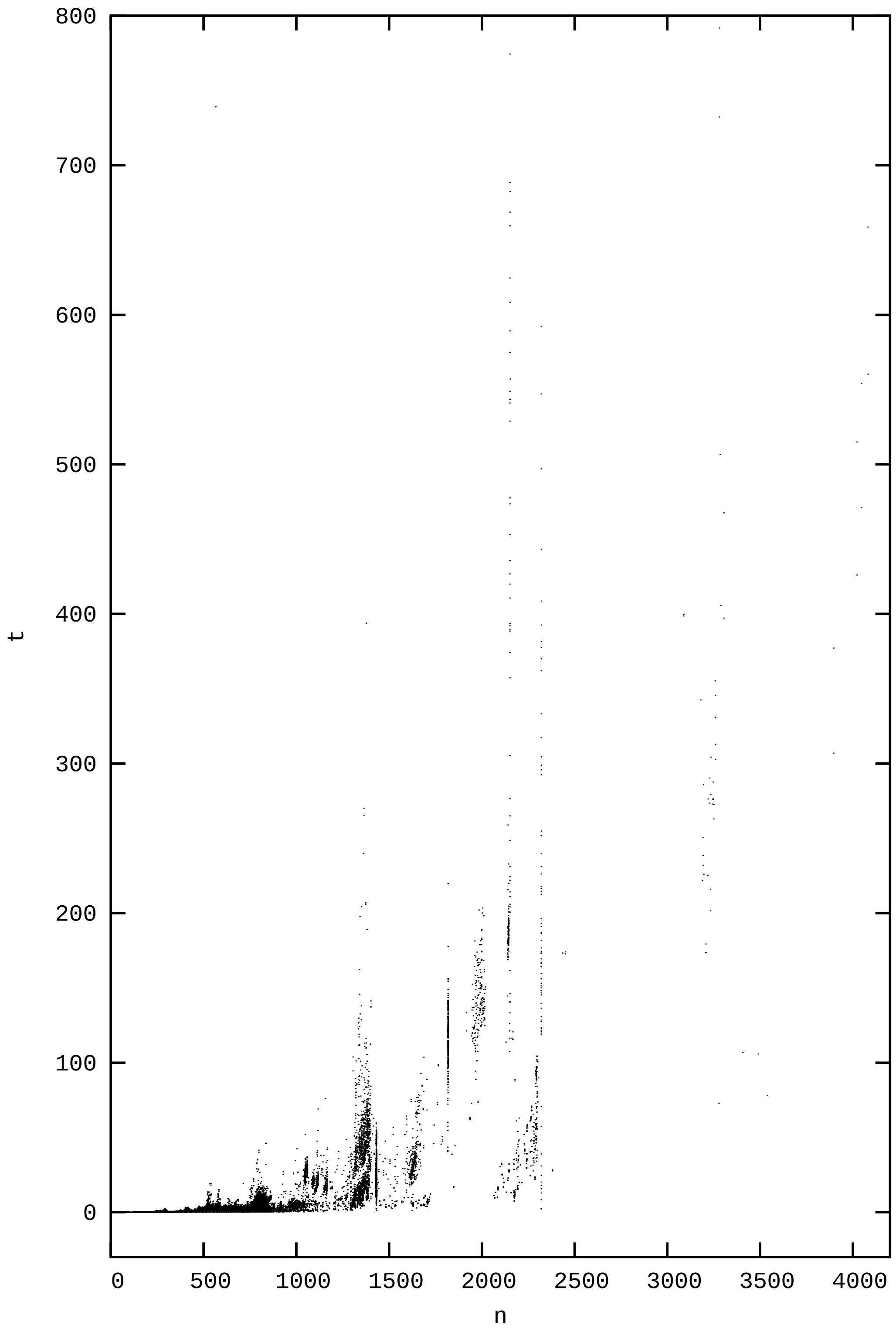


Figure A.19: Cactus cut-nodes based PB-heuristic: running time, $n := |V(G(x^*))|$

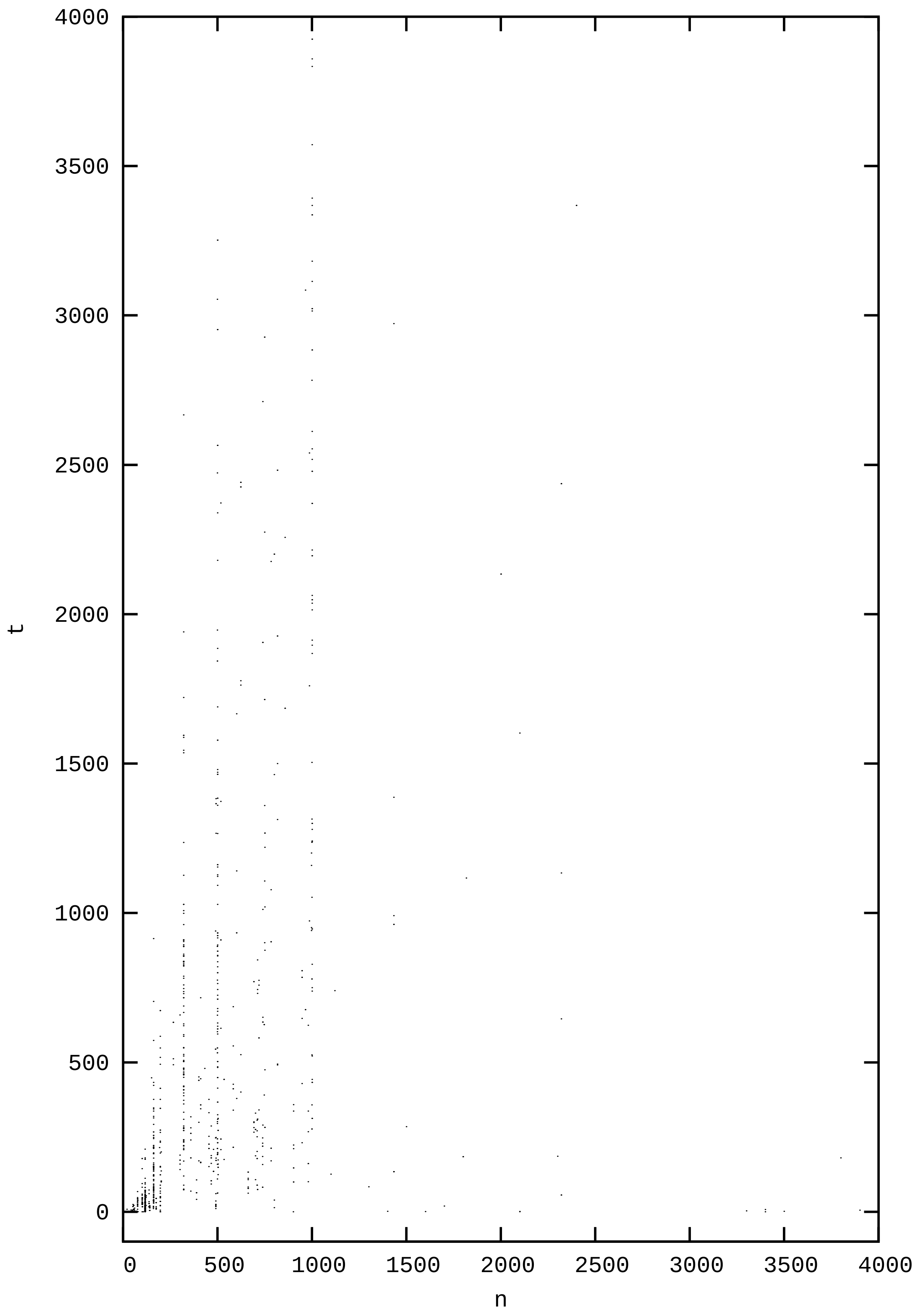


Figure A.20: Cactus cut-nodes based PB-heuristic: inequalities per call, $n := |V(G)|$

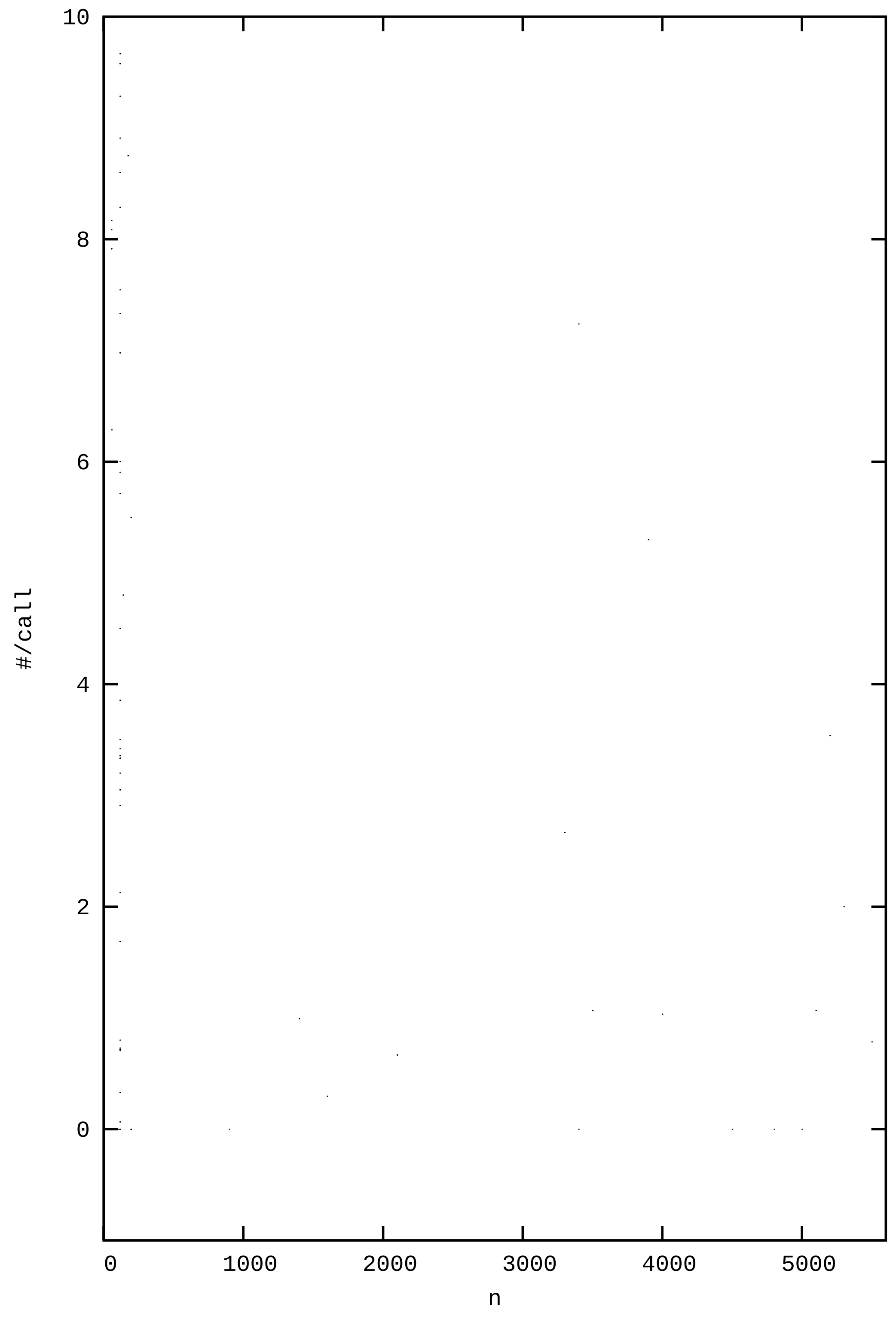


Figure A.21: Cactus cut-nodes based PB-heuristic: inequalities per call, zoomed, $n := |V(G)|$

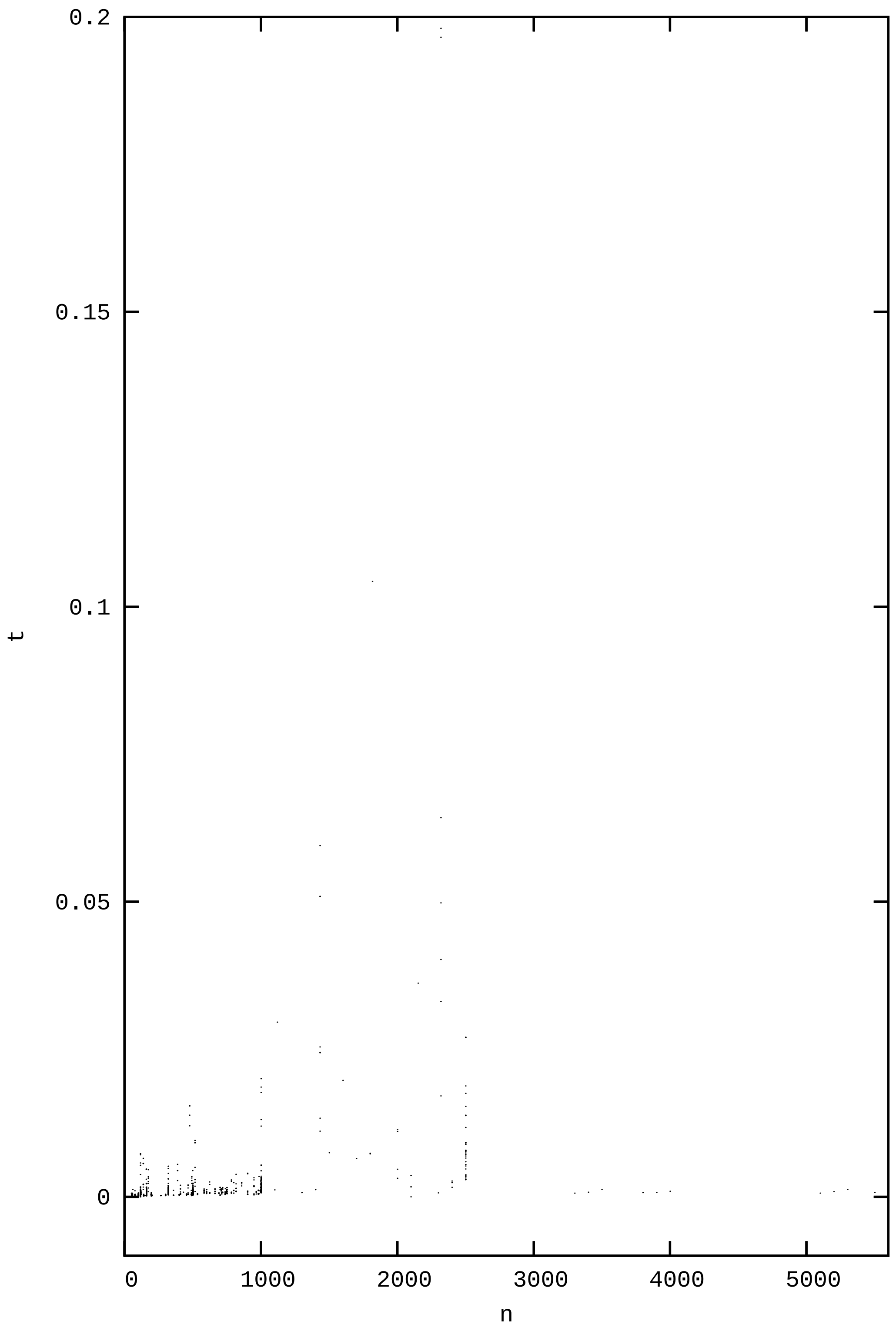


Figure A.22: Cactus cut-nodes based PB-heuristic: time per inequality, $n := |V(G)|$

Appendix B

Data of the instances

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
ALBA_3_1	116	66	174	5698	0.59	5732
ALBA_3_2	116	71	174	6704	0.17	6716
ALBA_3_3	116	72	174	6164	0.60	6201
ALBA_3_4	116	67	174	5924	0.03	5926
ALBA_3_5	116	62	174	5855	0.01	5856
ALBA_5_4	116	33	174	4707	0.25	4719

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
MADR_3_1	196	111	316	8621	0.68	8680
MADR_3_2	196	88	316	8140	0.18	8155
MADR_3_3	196	95	316	8490	0.76	8555
MADR_3_4	196	95	316	8648	0.37	8680
MADR_3_5	196	103	316	8728	0.30	8755
MADR_5_2	196	47	316	6535	0.38	6560
MADR_5_3	196	53	316	6893	0.89	6955
MADR_5_5	196	53	316	6730	0.89	6790

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
GRP04	116	48	174	4796	8.13	5186
GRP08	116	47	174	6799	0.22	6814
GRP10	116	48	174	4700	8.97	5122

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
WPP-A3101	116	44	174	5648	0.67	5686
WPP-A3112	116	44	174	19621	0.65	19750
WPP-A3201	116	44	174	5490	0.21	5502
WPP-A3205	116	44	174	5344	1.25	5411
WPP-A3207	116	44	174	3669	0.10	3673
WPP-A3209	116	44	174	6794	0.27	6813
WPP-A3211	116	44	174	15741	1.84	16031
WPP-A3212	116	44	174	15741	1.84	16031
WPP-A5103	116	44	174	5360	0.55	5390
WPP-A5108	116	44	174	3462	0.75	3488
WPP-A5109	116	44	174	7915	0.46	7952
WPP-A5152511	265	85	1136	89194	0.31	89473
WPP-A5205	116	44	174	5463	0.25	5477
WPP-A5208	116	44	174	3651	0.08	3654
WPP-A5209	116	44	174	7455	1.00	7530
WPP-A5211	116	44	174	16880	0.39	16947
WPP-A7101	116	44	174	5346	1.29	5415
WPP-A7103	116	44	174	5674	0.70	5714

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
WPP-A7104	116	44	174	5735	0.15	5744
WPP-A7107	116	44	174	3385	0.82	3413
WPP-A7109	116	44	174	8077	0.65	8130
WPP-A7112	116	44	174	16999	0.09	17015
WPP-A7205	116	44	174	5184	0.00	5184
WPP-A7206	116	44	174	5465	0.09	5470
WPP-A7207	116	44	174	3337	0.11	3341
WPP-A7208	116	44	174	3256	1.19	3295
WPP-A7210	116	44	174	7181	1.32	7276
WPP-A7212	116	44	174	16276	2.19	16633
WPP-B351	453	133	811	9206	1.01	9299
WPP-B352	453	133	811	9016	0.08	9024
WPP-B371	490	142	873	10208	0.46	10255
WPP-B372	490	142	873	10663	0.49	10716
WPP-B421	357	109	884	7483	0.65	7532
WPP-B422	357	109	884	7390	0.67	7440
WPP-B451	465	136	1055	8246	1.58	8377
WPP-B452	465	136	1055	8052	0.01	8053
WPP-B471	498	144	1114	8181	0.13	8192
WPP-B472	498	144	1114	8087	0.81	8153
WPP-B521	388	117	1106	7530	0.92	7600
WPP-B551	488	141	1318	7969	0.95	8045
WPP-B571	498	144	1326	7977	0.18	7992
WPP-B572	498	144	1326	8071	0.40	8104
WPP-B621	409	122	1392	7770	0.52	7811
WPP-B622	409	122	1392	7485	0.90	7553
WPP-B651	491	142	1532	7987	0.17	8001
WPP-B652	491	142	1532	7664	0.95	7737
WPP-B671	498	144	1537	8425	0.48	8466
WPP-B672	498	144	1537	8189	0.17	8203
WPP-C321	502	145	1013	9758	0.52	9809
WPP-C322	502	145	1013	10106	0.38	10145
WPP-C351	691	189	1236	11113	0.71	11193
WPP-C352	691	189	1236	10538	1.92	10741
WPP-C371	737	200	1319	11235	0.34	11274
WPP-C372	737	200	1319	11047	0.87	11144
WPP-C421	534	152	1339	8975	0.26	8999
WPP-C451	711	194	1605	10060	0.48	10109
WPP-C452	711	194	1605	9890	0.30	9920
WPP-C471	746	202	1693	10328	0.61	10392
WPP-C521	582	164	1705	8994	0.23	9015
WPP-C522	582	164	1705	8887	0.15	8901
WPP-C551	718	196	1920	9292	1.17	9401
WPP-C552	718	196	1920	9422	0.61	9480
WPP-C571	749	203	2005	9925	0.36	9961
WPP-C572	749	203	2005	9638	0.22	9660
WPP-C621	622	173	2080	8850	0.47	8892
WPP-C622	622	173	2080	8833	0.27	8857
WPP-C651	739	200	2288	9168	0.30	9196
WPP-C652	739	200	2288	9124	0.16	9139
WPP-C671	750	203	2269	9444	0.51	9493
WPP-C672	750	203	2269	9248	0.15	9262

name	$\|V(G)\|$	$\|\mathcal{C}\|$	$\|E(G)\|$	GL-lb	GL-gap	best ub
WPP-D321	661	182	1297	10594	0.74	10673
WPP-D322	661	182	1297	11240	0.32	11276
WPP-D351	902	238	1611	13045	0.13	13063
WPP-D352	902	238	1611	13001	0.14	13020
WPP-D371	979	255	1738	12647	0.46	12706
WPP-D372	979	255	1738	12615	0.07	12625
WPP-D421	708	193	1867	9581	0.25	9605
WPP-D422	708	193	1867	9454	0.71	9522
WPP-D451	947	248	2161	10990	1.10	11111
WPP-D452	947	248	2161	10965	0.10	10977
WPP-D471	996	258	2182	11450	0.48	11506
WPP-D472	996	258	2182	11301	58.38	17899
WPP-D521	783	211	2361	9509	0.22	9530
WPP-D522	783	211	2361	9826	0.85	9910
WPP-D551	965	252	2643	10814	0.12	10827
WPP-D552	965	252	2643	10817	56.05	16880
WPP-D571	999	259	2678	10759	0.26	10788
WPP-D572	999	259	2678	10602	0.52	10658
WPP-D621	817	218	2793	10064	0.13	10078
WPP-D622	817	218	2793	9985	0.22	10007
WPP-D651	985	256	3036	10551	49.90	15817
WPP-D652	985	256	3036	10577	70.01	17982
WPP-D671	999	259	3073	10902	61.87	17648
WPP-D672	999	259	3073	10520	0.04	10525

name	$\|V(G)\|$	$\|\mathcal{C}\|$	$\|E(G)\|$	GL-lb	GL-gap	best ub
dv160.rq.1_24	160	130	240	26553	0.18	26601
dv160.rq.1_30	160	120	320	17200	0.24	17242
dv160.rq.1_31	160	120	320	18369	0.01	18371
dv160.rq.1_33	160	120	320	17486	0.38	17553
dv160.rq.1_34	160	120	320	17241	0.00	17242
dv160.rq.1_47	160	130	240	22591	0.04	22601
dv160.rq.1_48	160	130	240	22317	0.02	22322
dv160.rq.1_49	160	130	240	21803	0.25	21859
dv160.rq.1_55	160	120	320	15859	0.01	15862
dv160.rq.1_57	160	120	320	16201	0.30	16251
dv160.rq.1_59	160	120	320	15547	0.04	15554
dv160.rq.1_5	160	120	320	21668	0.06	21682
dv160.rq.1_7	160	120	320	21841	0.01	21845
dv160.rq.1_81	160	120	320	15678	0.04	15685
dv160.rq.1_95	160	130	240	18374	0.00	18375
dv320.rq.1_16	320	265	480	53523	0.01	53531
dv320.rq.1_18	320	265	480	52285	0.08	52332
dv320.rq.1_19	320	265	480	53372	0.14	53449
dv320.rq.1_26	320	252	640	34534	0.02	34543
dv320.rq.1_28	320	252	640	35045	0.08	35076
dv320.rq.1_36	320	265	480	43530	0.32	43670
dv320.rq.1_38	320	265	480	42174	0.18	42252
dv320.rq.1_48	320	252	640	31123	0.02	31130
dv320.rq.1_49	320	252	640	30860	0.06	30880
dv320.rq.1_55	320	265	480	37693	0.15	37752
dv320.rq.1_56	320	265	480	39091	0.06	39116
dv320.rq.1_58	320	265	480	39691	0.11	39735
dv320.rq.1_59	320	265	480	36727	0.14	36782
dv320.rq.1_5	320	252	640	43177	0.00	43178

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
dv320.rq.1_65	320	252	640	31203	0.01	31208
dv320.rq.1_66	320	252	640	30844	0.07	30866
dv320.rq.1_75	320	265	480	37585	0.14	37638
dv320.rq.1_77	320	265	480	38005	0.01	38011
dv320.rq.1_79	320	265	480	39498	0.04	39517
dv320.rq.1_7	320	252	640	41237	0.01	41245
dv320.rq.1_8	320	252	640	42154	0.13	42212
dv320.rq.1_9	320	252	640	41673	0.00	41675

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
steinb08.rq.066	75	67	94	545	0.18	546
steinb11.rq.066	75	65	150	279	0.35	280
steinb13.rq.066	100	90	125	719	0.00	719
steinb13.rq.1	100	86	125	713	0.42	716
steinb15.rq.066	100	90	125	688	0.29	690
steinb17.rq.1	100	76	200	376	0.53	378
steinb18.rq.1	100	76	200	416	0.72	419
steinc01.rq.066	500	459	625	3755	0.07	3758
steinc01.rq.1	500	435	625	3616	0.11	3620
steinc02.rq.066	500	459	625	3506	0.14	3511
steinc02.rq.1	500	435	625	3360	0.20	3367
steinc03.rq.066	500	459	625	3551	0.02	3552
steinc03.rq.1	500	435	625	3447	0.05	3449
steinc04.rq.066	500	459	625	3618	0.11	3622
steinc04.rq.1	500	435	625	3514	0.11	3518
steinc05.rq.066	500	459	625	3663	0.08	3666
steinc05.rq.1	500	435	625	3556	0.16	3562
steinc06.rq.066	500	430	1000	2215	0.13	2218
steinc07.rq.066	500	430	1000	2245	0.75	2262
steinc08.rq.066	500	430	1000	2156	0.00	2156
steinc08.rq.1	500	396	1000	2091	0.14	2094
steinc09.rq.1	500	396	1000	1903	0.21	1907
steinc10.rq.1	500	396	1000	2075	0.04	2076
steind01.rq.066	1000	913	1250	6886	0.07	6891
steind01.rq.1	1000	878	1250	6777	0.04	6780
steind02.rq.066	1000	913	1250	6979	0.11	6987
steind02.rq.1	1000	878	1250	6813	0.10	6820
steind03.rq.066	1000	913	1250	7333	0.01	7334
steind03.rq.1	1000	878	1250	7228	0.04	7231
steind04.rq.066	1000	913	1250	6983	0.14	6993
steind05.rq.066	1000	913	1250	7369	0.10	7377
steind05.rq.1	1000	878	1250	7234	0.17	7247
steind06.rq.066	1000	859	2000	4374	0.11	4379
steind06.rq.1	1000	807	2000	4257	0.11	4262
steind07.rq.066	1000	859	2000	4400	0.02	4401
steind07.rq.1	1000	807	2000	4217	0.02	4218
steind08.rq.066	1000	859	2000	4261	0.14	4267
steind09.rq.066	1000	859	2000	4394	0.09	4398
steind09.rq.1	1000	807	2000	4194	33.21	5587
steind10.rq.066	1000	859	2000	4373	0.04	4375
steind10.rq.1	1000	807	2000	4147	0.04	4149
steine01.rq.066	2500	2286	3125	17339	0.14	17365
steine01.rq.1	2500	2204	3125	16985	13.77	19324

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
steine02.rq.066	2500	2286	3125	18261	0.12	18284
steine02.rq.1	2500	2204	3125	17848	0.07	17862
steine03.rq.066	2500	2286	3125	18243	13.83	20767
steine03.rq.1	2500	2204	3125	17887	0.09	17904
steine04.rq.066	2500	2286	3125	17903	0.15	17930
steine04.rq.1	2500	2204	3125	17603	0.11	17624
steine05.rq.066	2500	2286	3125	18513	0.09	18531
steine05.rq.1	2500	2204	3125	18083	0.07	18097

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
vrpfeas1.infty-metr.rq.1	51	28	187	244	0.81	246
vrpfeas3.infty-metr.rq.1	101	57	387	370	0.00	370
vrpfeas6.infty-metr.rq.1	120	70	461	306	0.65	308
vrpfeas7.infty-metr.rq.1	101	55	379	266	1.12	269
vrpfeas8.infty-metr.rq.1	857	542	3403	11780	0.47	11836

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
egl-e1-A	77	27	98	1370	1.16	1386
egl-e2-A	77	11	98	1082	0.27	1085
egl-s1-A	140	66	190	2324	0.81	2343

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
uni1.1.1-metr_0	55	49	78	611304	23.21	753237
uni1.1.1-metr_2	2152	2152	4311	6619256	48.76	9847250
uni1.1.1-metr_3	1120	569	7582	4270944	70.48	7281269
uni1.1.1-metr_4	1416	693	5122	3037489	81.58	5515489
uni1.1.infty-metr_0	55	49	165	539329	12.30	605683
uni1.1.infty-metr_2	2152	2152	16000	6352333	55.75	9894261
uni1.1.infty-metr_3	1120	569	8350	3110545	74.03	5413426
uni1.1.infty-metr_4	1416	693	10757	2368218	89.47	4487088
uni2.1.1-metr_4	136	136	367	2121322	1.65	2156454
uni2.1.1-metr_6	477	477	2261	3897829	40.88	5491462
uni2.1.1-metr_8	1817	1817	3652	6003737	45.12	8713112
uni2.1.infty-metr_4	136	136	912	1825313	1.02	1844047
uni2.1.infty-metr_6	477	477	3484	2875126	0.71	2895611
uni2.1.infty-metr_8	1817	1817	13916	5600121	41.86	7944401
uni3.1.1-metr_2	516	258	2362	4199913	0.00	4200200
uni3.1.1-metr_3	174	87	396	3245000	0.77	3270000
uni3.1.1-metr_4	1432	1432	2838	15860000	0.37	15920000
uni3.1.infty-metr_2	516	258	3602	3117825	0.50	3133450
uni3.1.infty-metr_3	174	87	867	2465000	0.20	2470000
uni3.1.infty-metr_4	1432	1432	6146	14795000	0.10	14810000
uni4.1.1-metr_0	159	159	448	4920000	0.81	4960000
uni4.1.1-metr_1	2319	2319	4539	23650000	0.04	23660000
uni4.1.infty-metr_1	2319	2319	8830	23280000	0.00	23280000
uni5.1.1-metr_0	1184	584	6339	14608108	85.80	27143193
uni5.1.1-metr_1	1060	1060	5315	27303250	51.19	41279940
uni5.1.infty-metr_0	1184	584	13024	10558502	91.27	20195764
uni5.1.infty-metr_1	1060	1060	12074	19589275	47.90	28974319

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
alb1000.c0-r0-rq.1	1000	807	1998	2541	0.11	2544
alb1000.c1-r2-rq.15	1000	697	1998	1912	0.15	1915
alb2000.c0-r0-rq.1	2000	1630	3996	5264	0.01	5265
alb2000.c1-r2-rq.15	2000	1402	3996	3896	0.12	3901

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
h1.n0100.rq.125	100	43	529	41878	0.53	42100
h1.n0200.rq.125	200	96	873	74986	0.20	75142

name	$\lvert V(G)\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert E(G)\rvert$	GL-lb	GL-gap	best ub
h1.n0300.rq.125	300	155	1260	89661	0.05	89710
h1.n0400.rq.125	400	173	1795	90268	0.18	90439
h1.n0500.rq.125	500	180	2807	94919	0.31	95222
h1.n0600.rq.125	600	201	3235	101576	0.33	101914
h1.n0700.rq.125	700	116	5507	82750	0.37	83059
h1.n0800.rq.125	800	268	4604	116764	0.59	117453
h1.n0900.rq.125	900	375	4662	127225	62.28	206463
h1.n1000.rq.125	1000	338	5756	127770	0.09	127885
h1.n1100.rq.125	1100	320	6967	124798	0.35	125242
h1.n1200.rq.125	1200	568	5306	158550	69.50	268746
h1.n1300.rq.125	1300	582	6228	155803	74.68	272164
h1.n1400.rq.125	1400	588	7224	157020	63.34	256477
h1.n1500.rq.125	1500	572	8294	160960	79.83	289466
h1.n1600.rq.125	1600	710	7548	178032	65.89	295341

Appendix C

Figures for Chapter 12

C.1 Sizes and Ghiani-Laporte gap

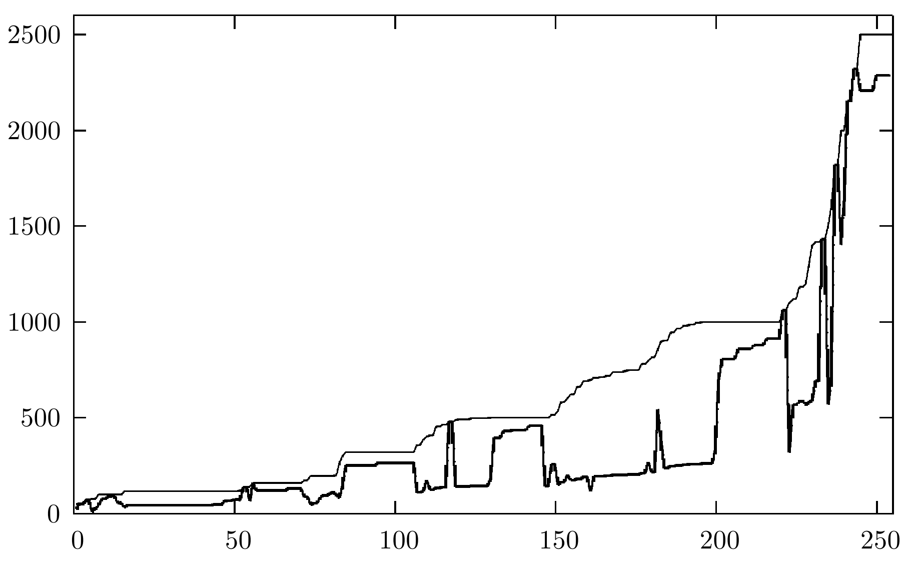


Figure C.1: Sizes of the instances: n and $|\mathcal{C}|$

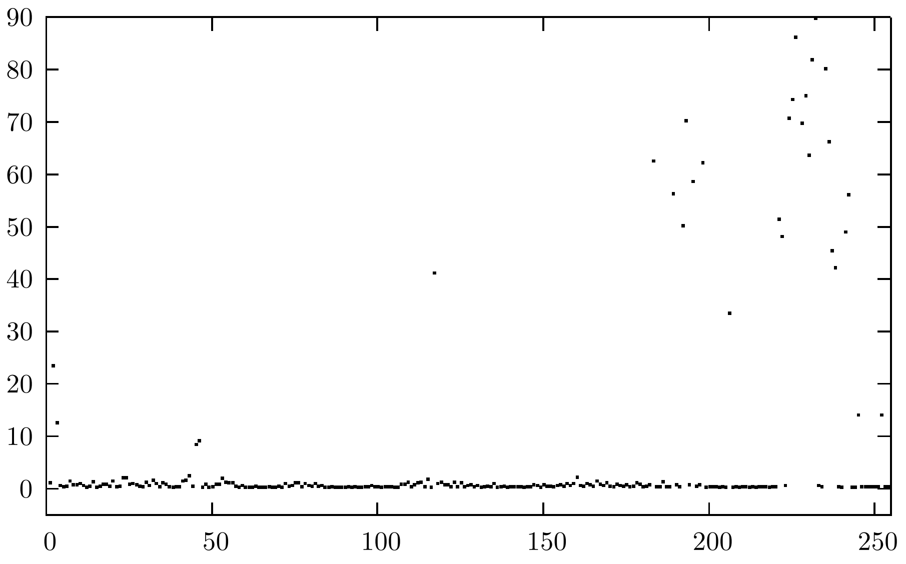


Figure C.2: Ghiani-Laporte gap for fixed tree

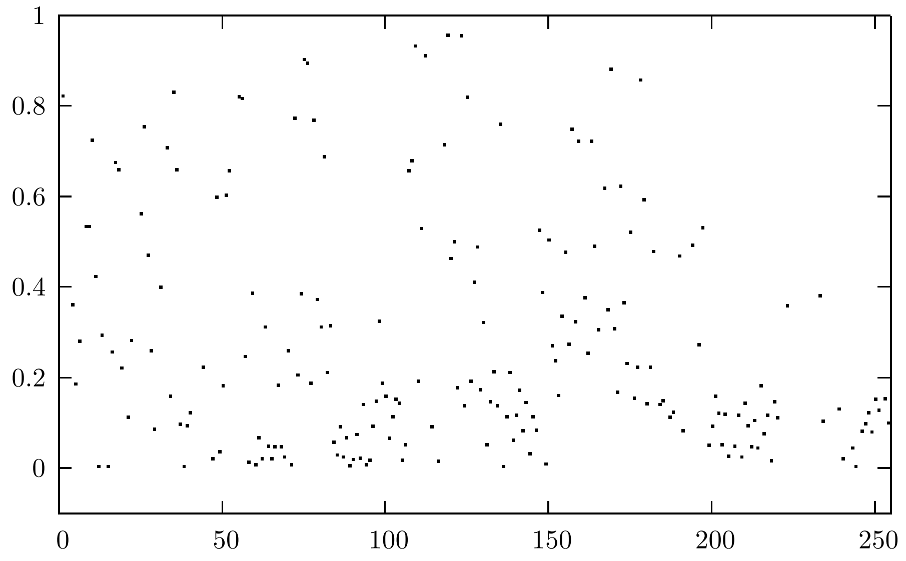


Figure C.3: Ghiani-Laporte gap for fixed tree, zoomed

C.2 Gap closures

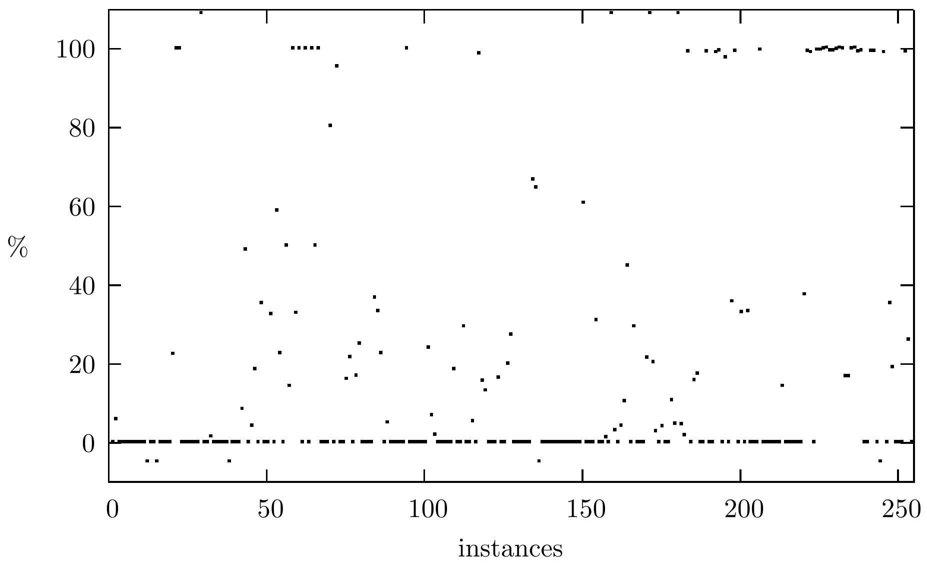


Figure C.4: KC-cactus heuristic: Gap closure

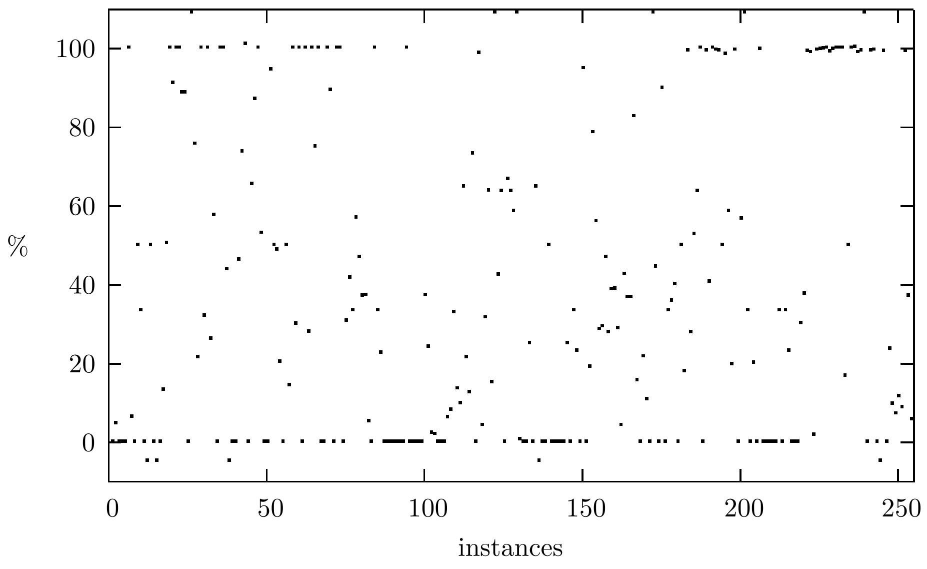


Figure C.5: Path-finder heuristic: Gap closure

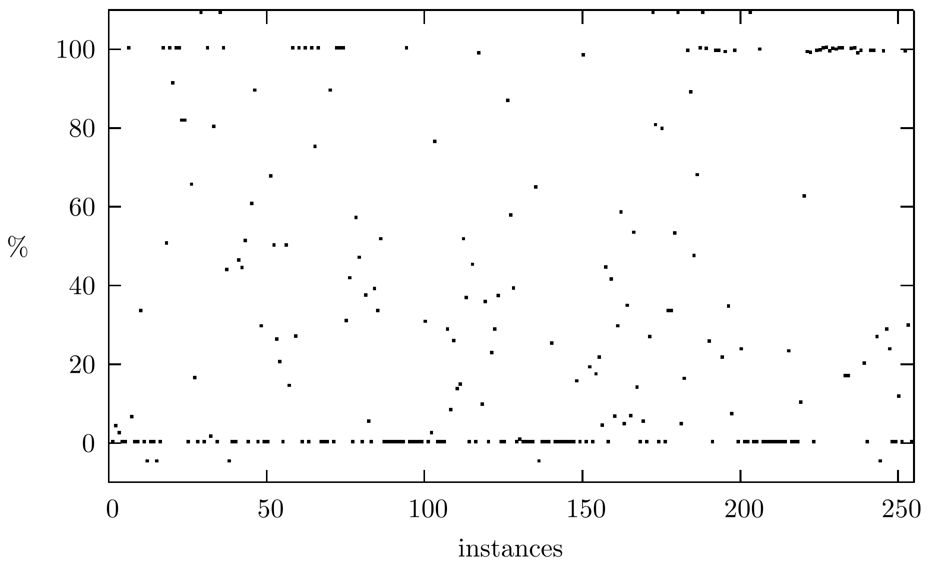


Figure C.6: Cactus based PB heuristic: Gap closure

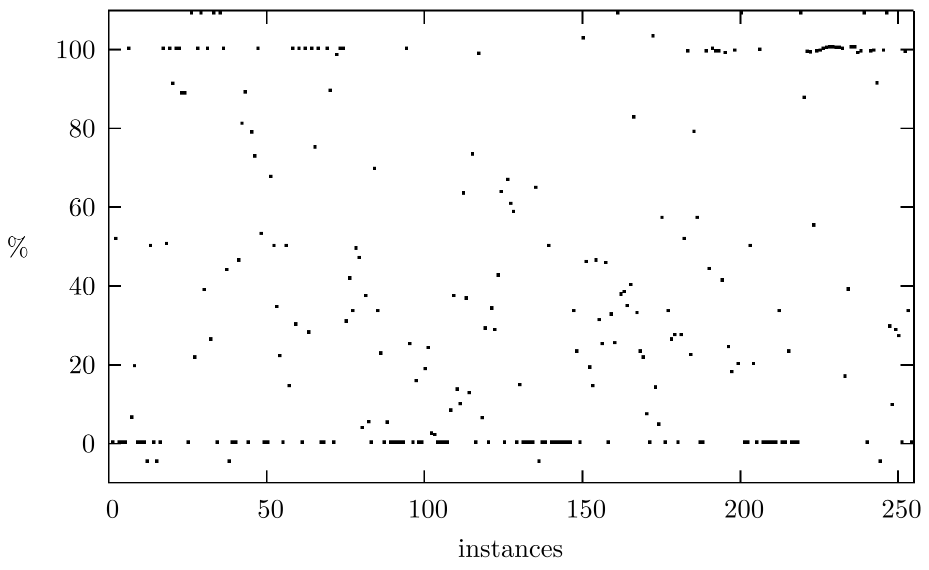


Figure C.7: Cactus cut-nodes PB heuristic: Gap closure

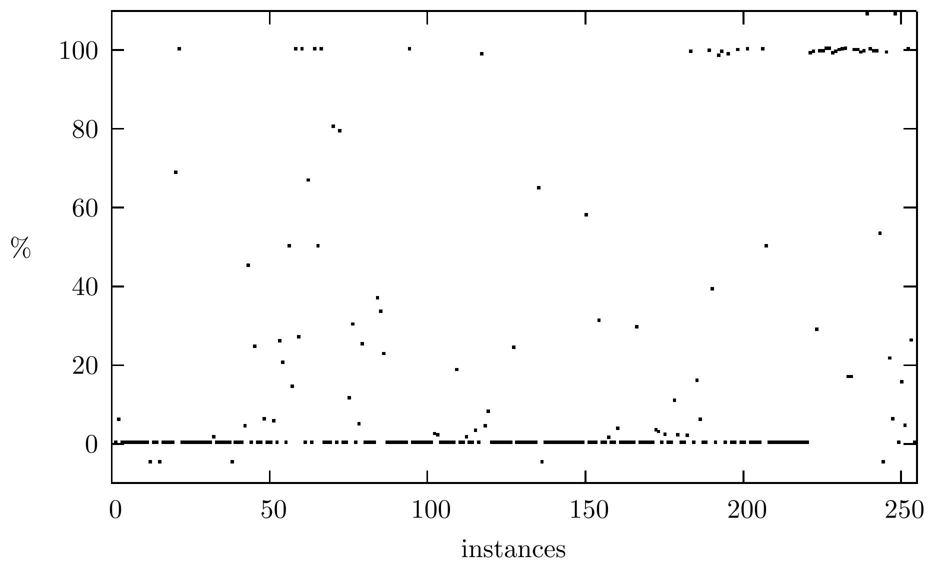


Figure C.8: All heuristics: Gap closure

C.3 Relative changes of lower bounds

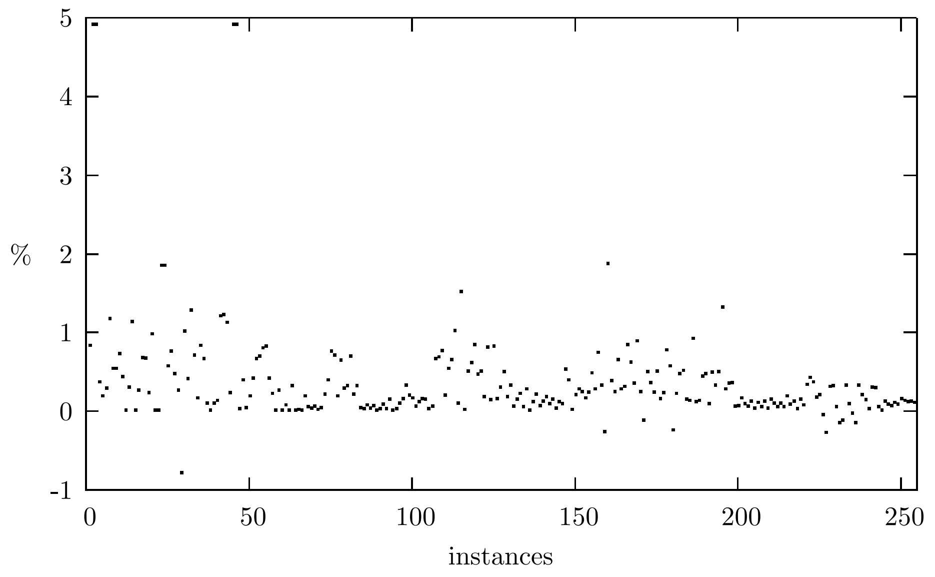


Figure C.9: KC-cactus heuristic: Relative change of lower bound

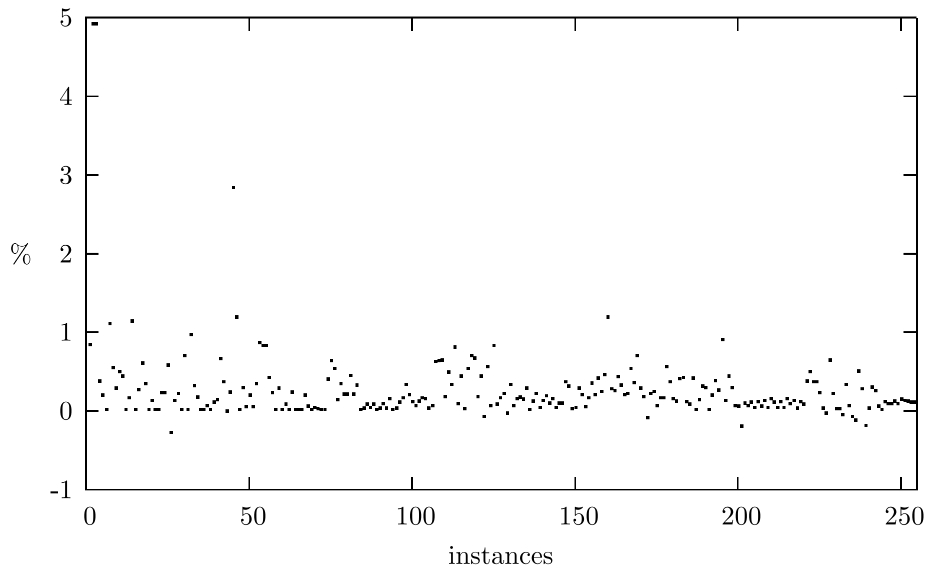


Figure C.10: Path-finder heuristic: Relative change of lower bound

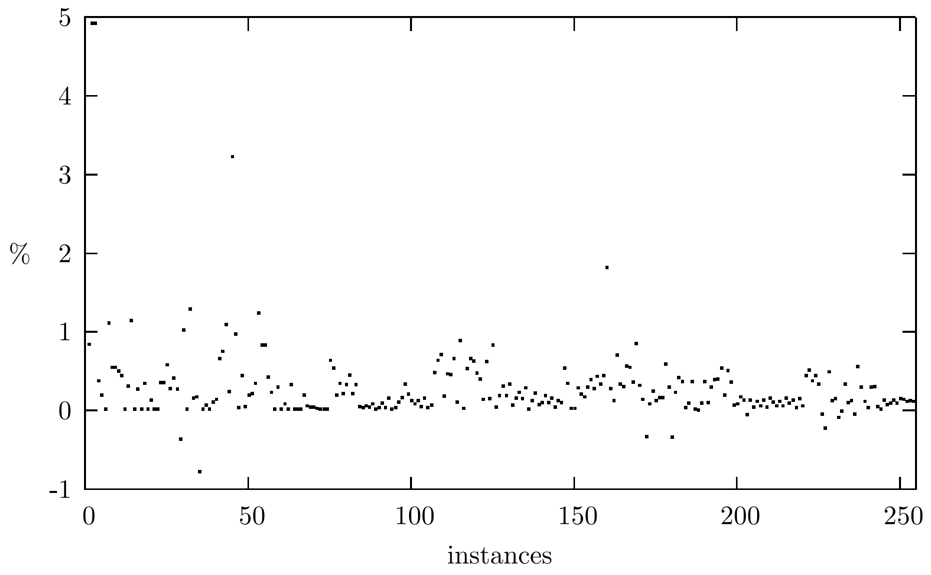


Figure C.11: Cactus based PB heuristic: Relative change of lower bound

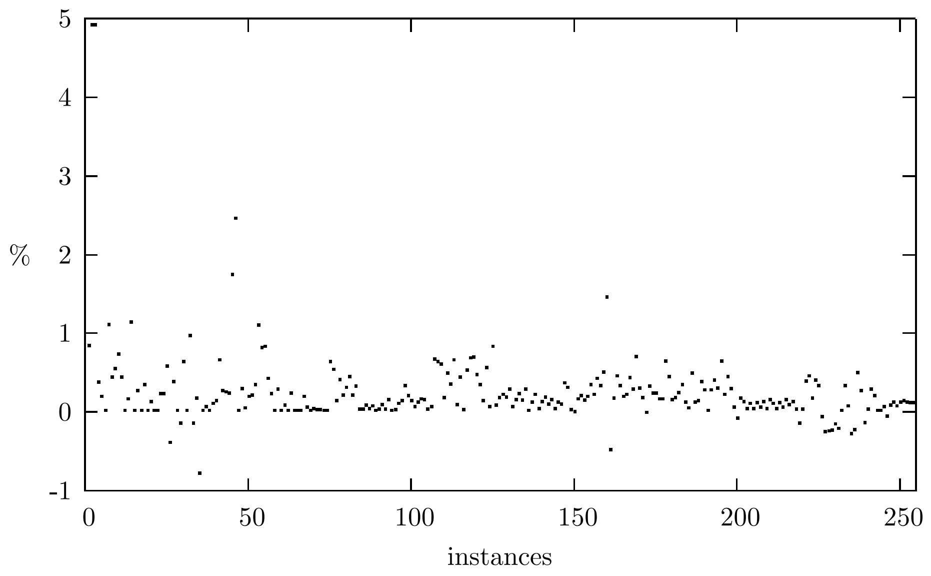


Figure C.12: Cactus cut-nodes PB heuristic: Relative change of lower bound

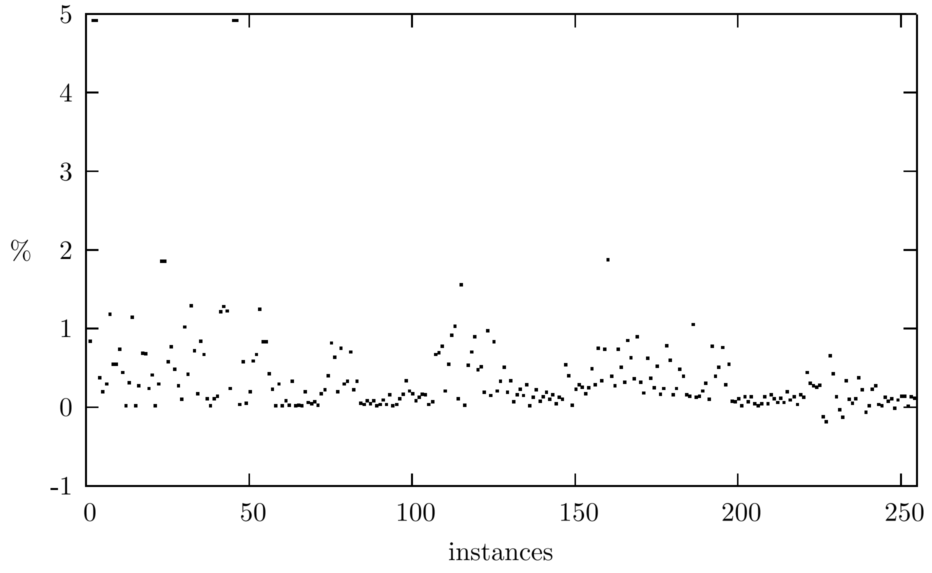


Figure C.13: All heuristics: Relative change of lower bound

C.4 Total running times

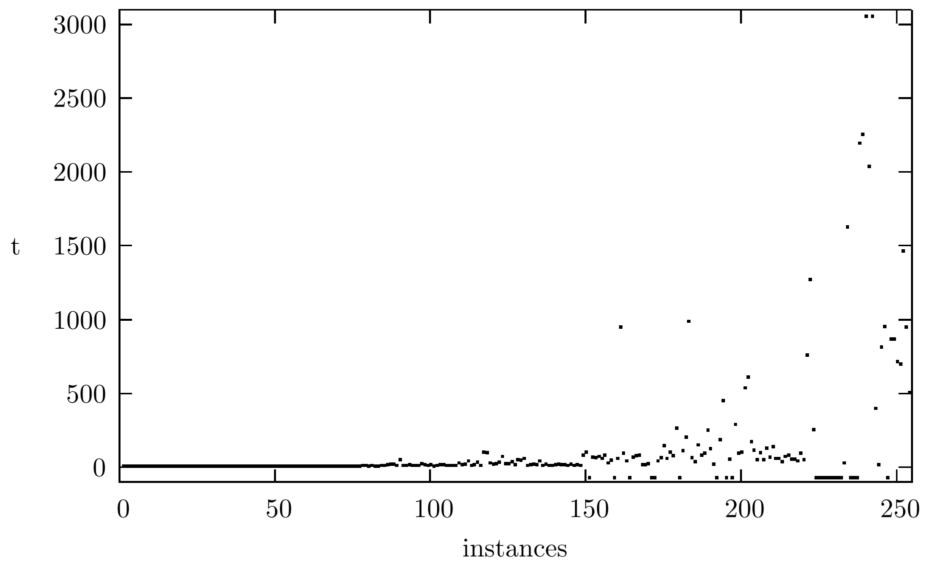


Figure C.14: KC-cactus heuristic: Total running times

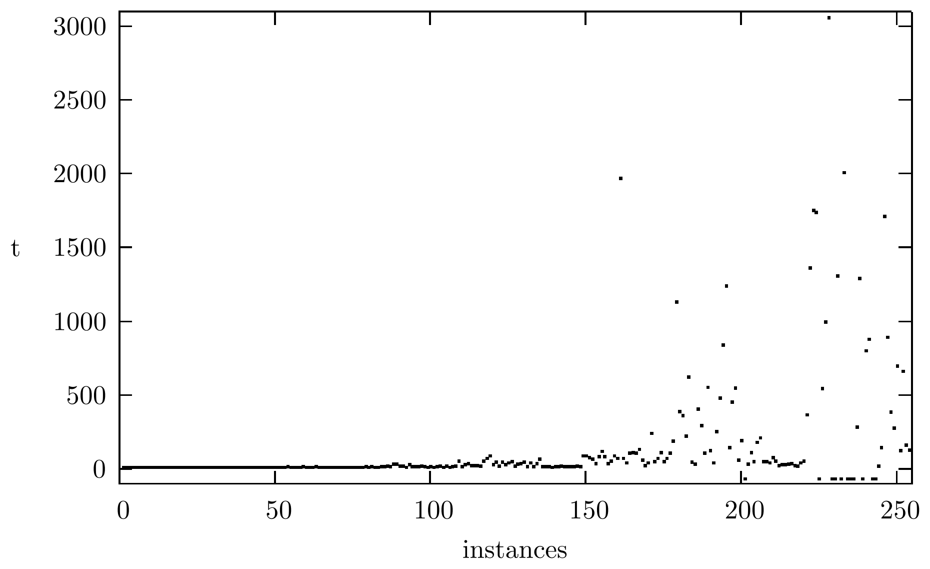


Figure C.15: Path-finder heuristic: Total running times

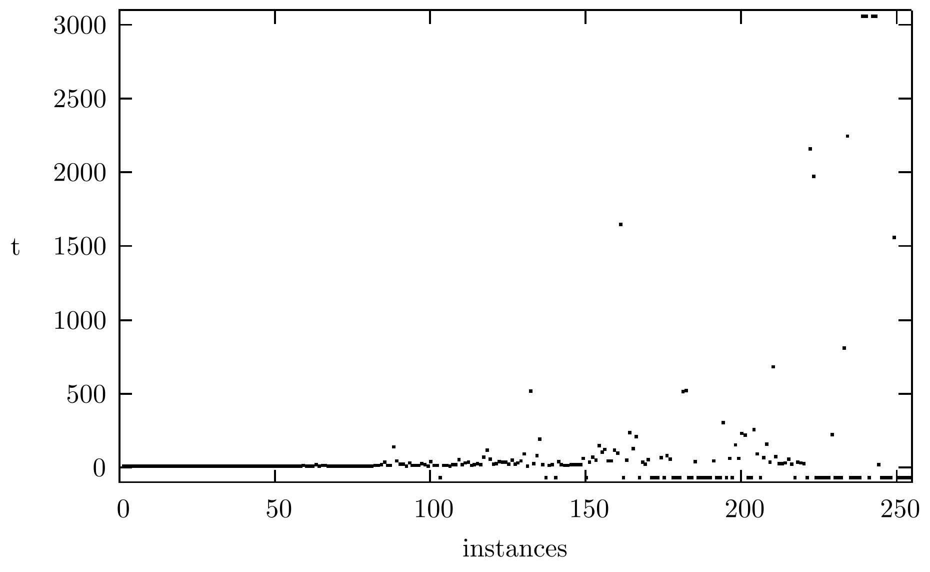


Figure C.16: Cactus based PB heuristic: Total running times

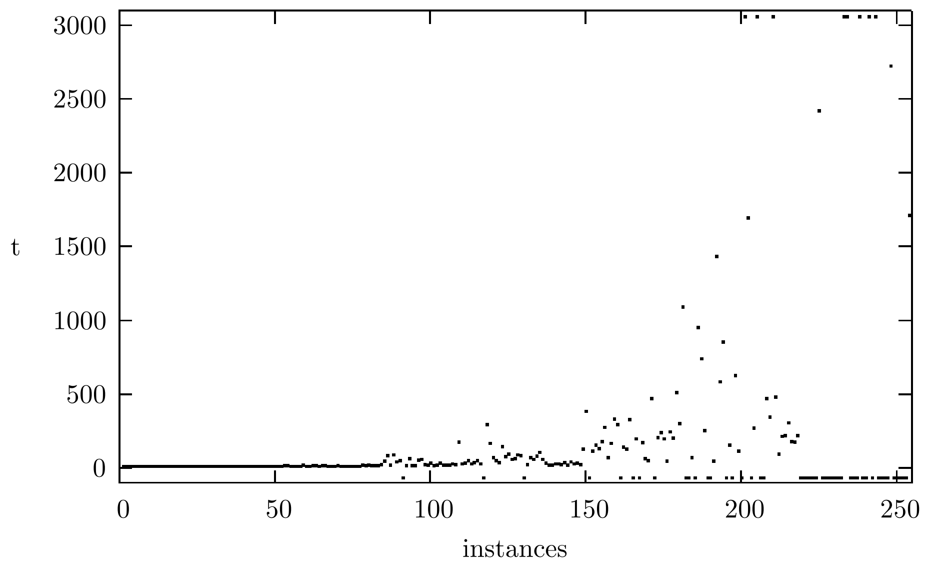


Figure C.17: Cactus cut-nodes PB heuristic: Total running times

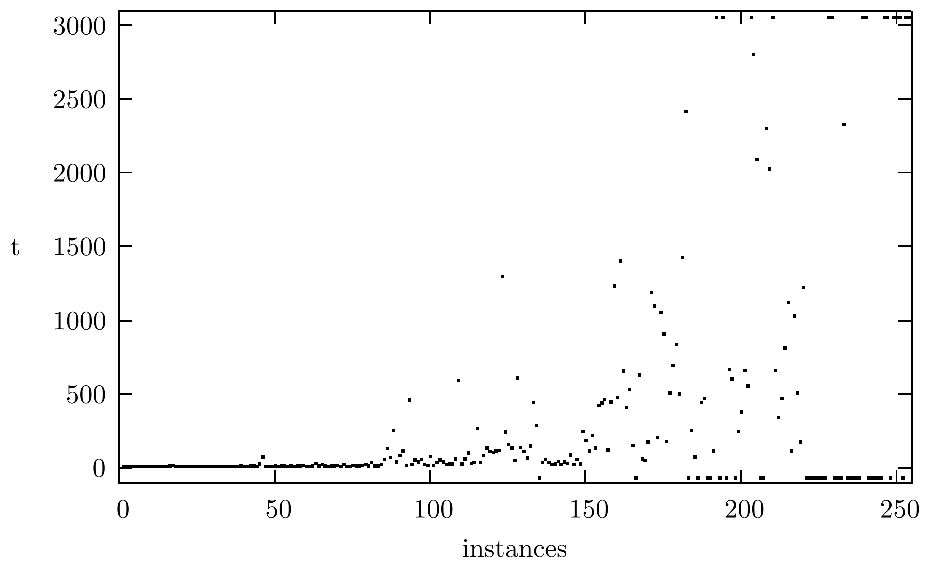


Figure C.18: All heuristics: Total running times

C.5 Number of LPs

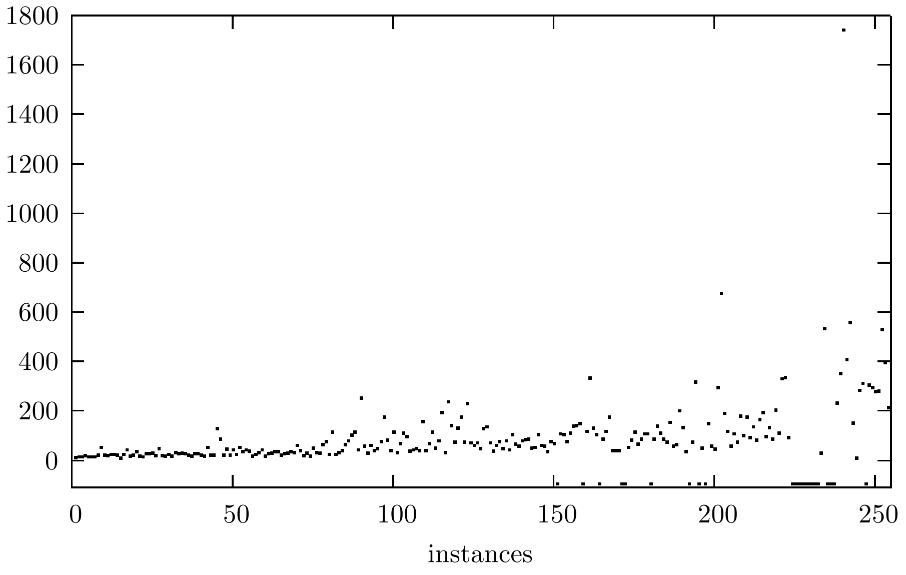


Figure C.19: KC-cactus heuristic: Number of LPs

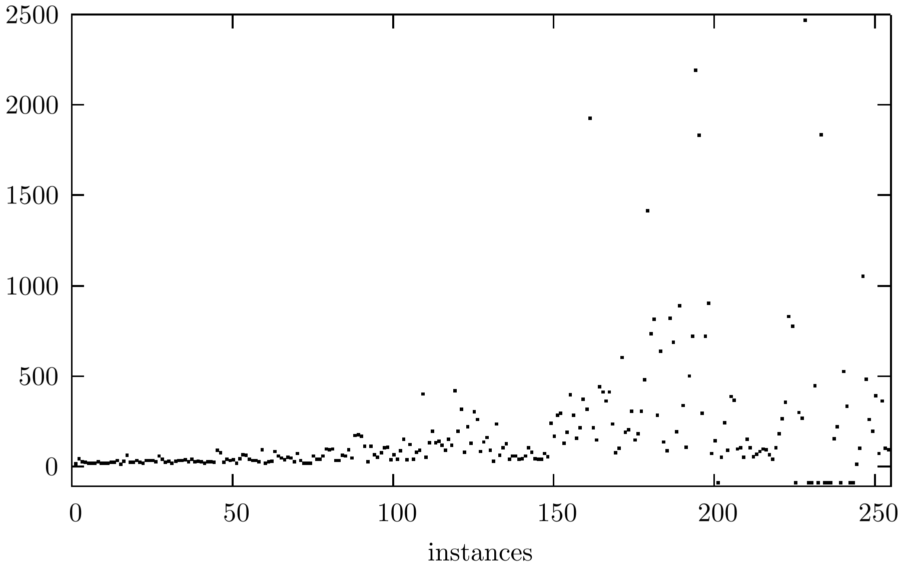


Figure C.20: Path-finder heuristic: Number of LPs

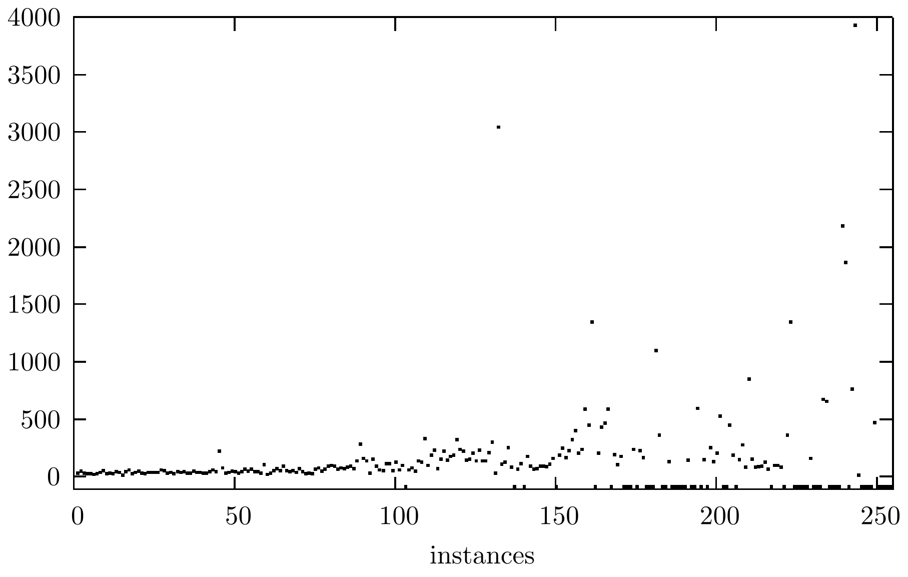


Figure C.21: Cactus based PB heuristic: Number of LPs

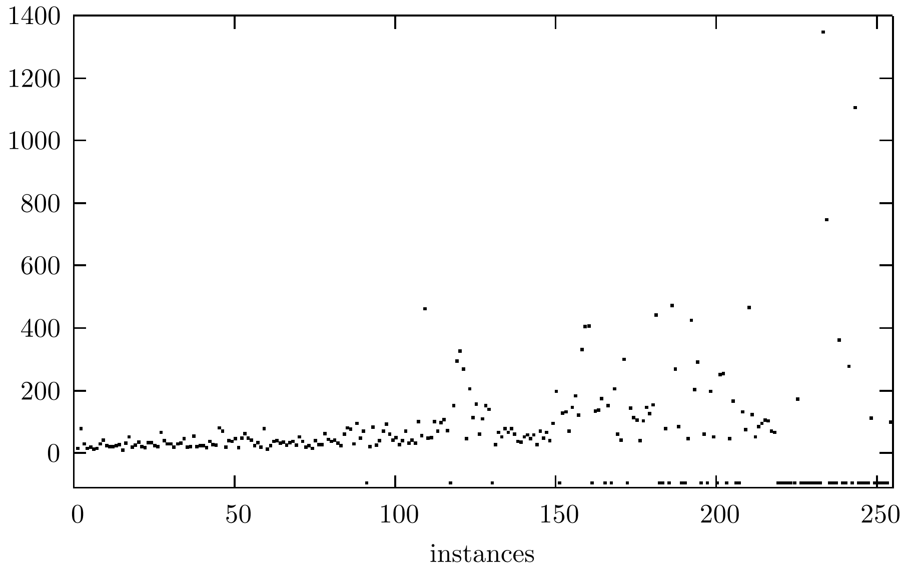


Figure C.22: Cactus cut-nodes PB heuristic: Number of LPs

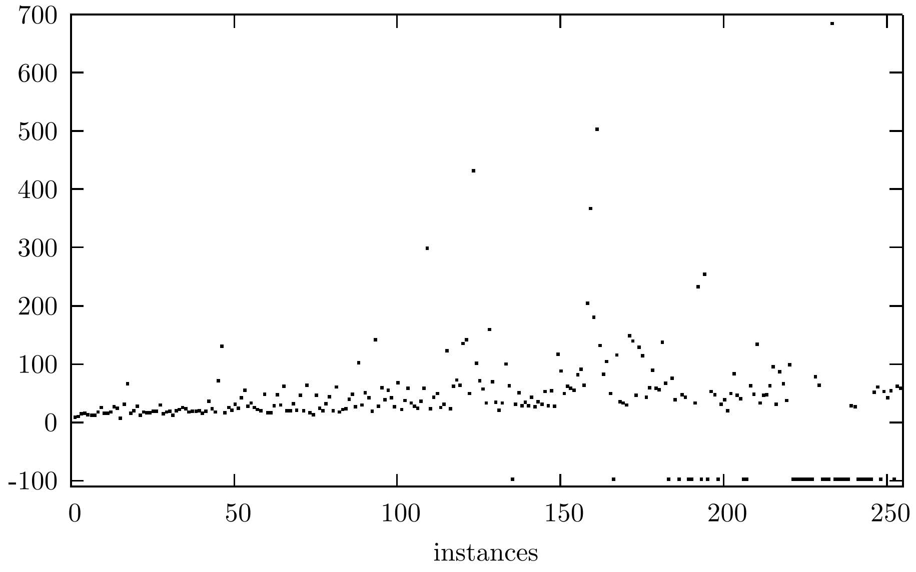


Figure C.23: All heuristics: Number of LPs

C.6 Solution by Branch-and-Cut

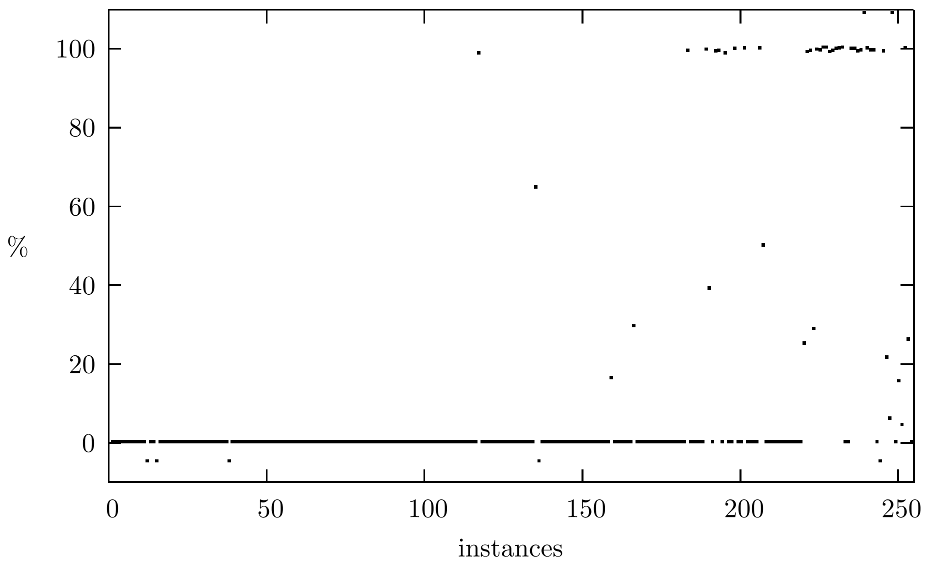


Figure C.24: Gap closure

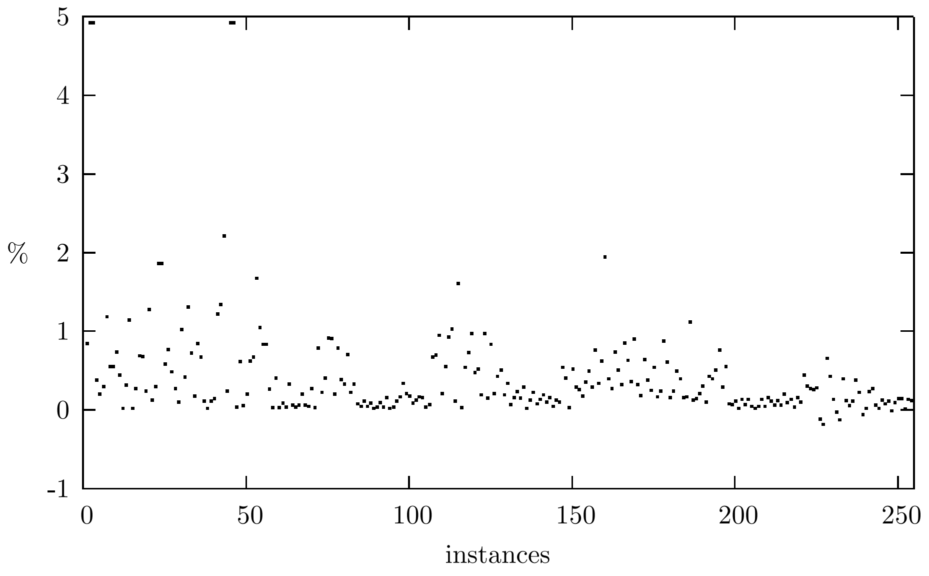


Figure C.25: Relative change of lower bound

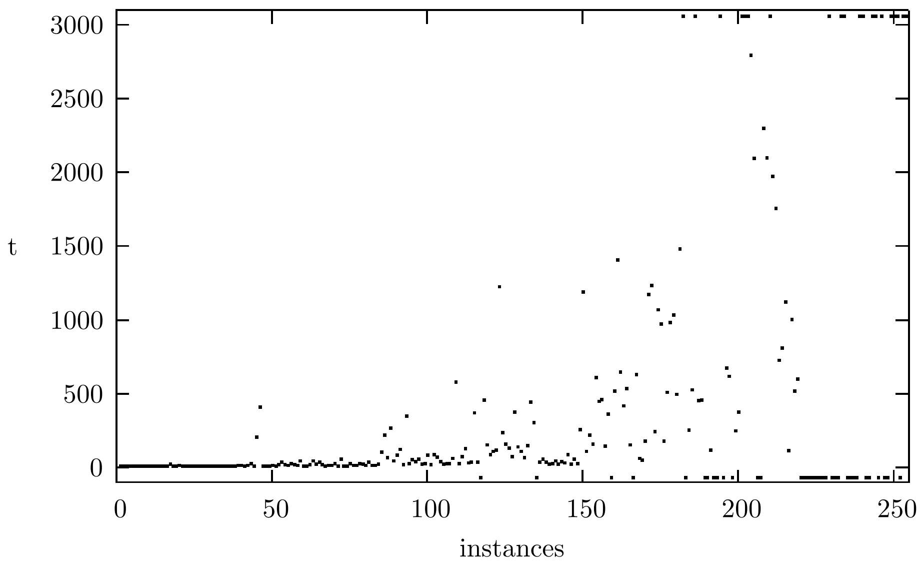


Figure C.26: Solution times $t \leq 3000s$

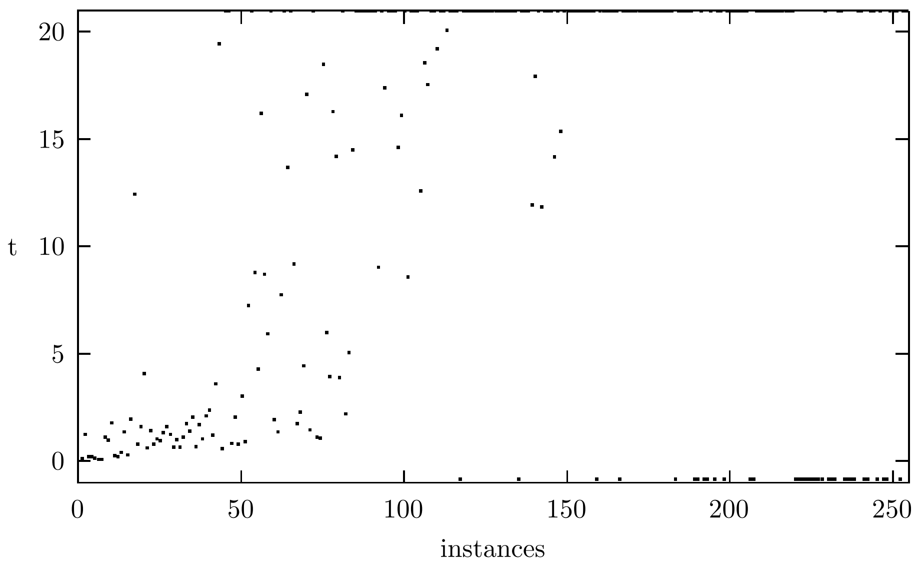


Figure C.27: Solution times $t \leq 20s$

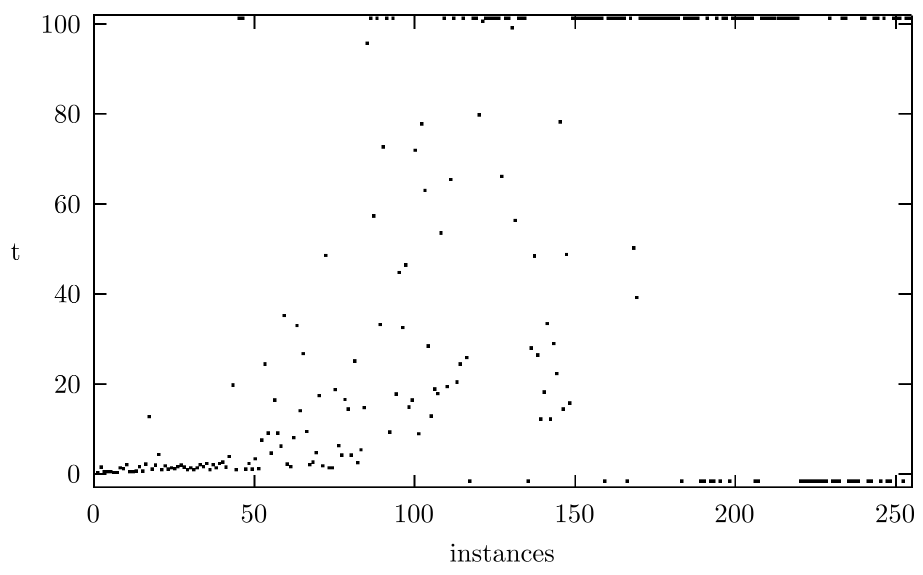


Figure C.28: Solution times $t \leq 100s$

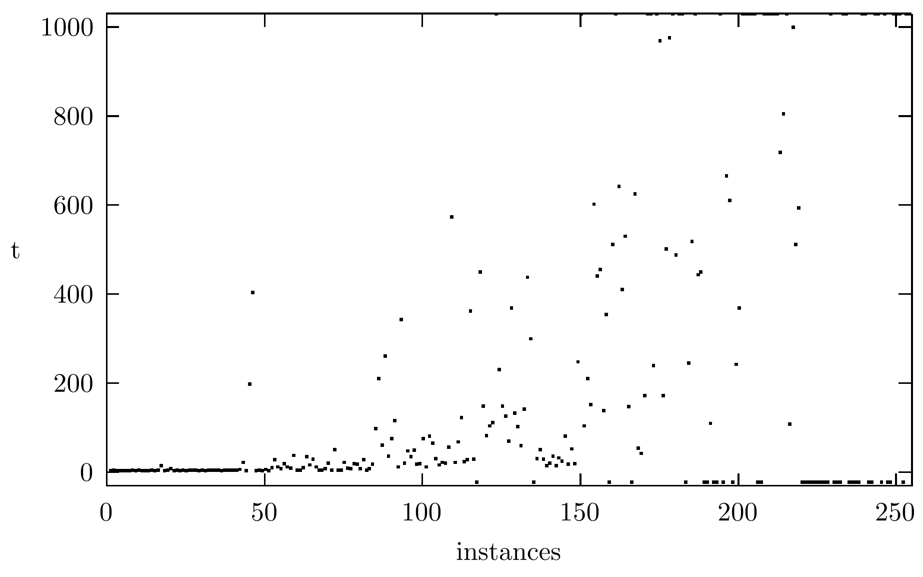


Figure C.29: Solution times $t \leq 1000s$

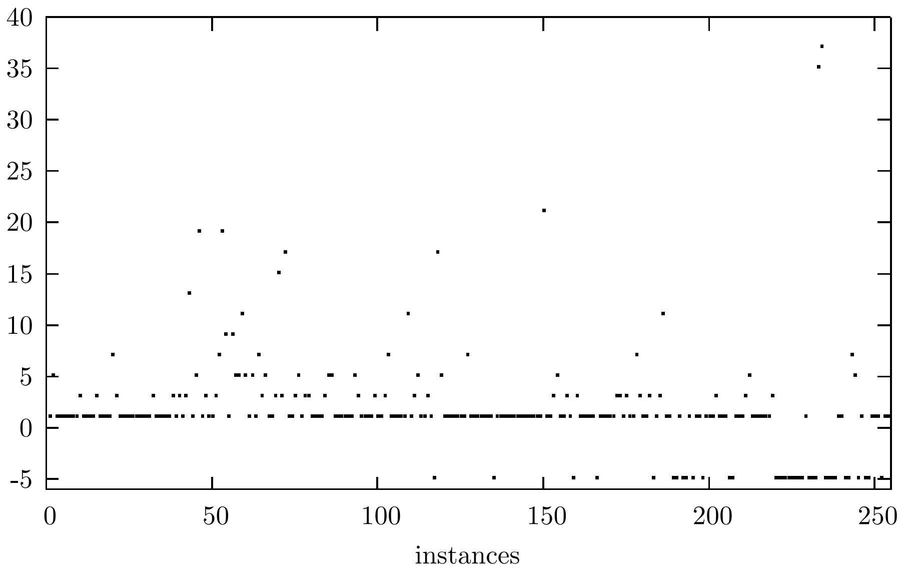


Figure C.30: Total number of subproblems

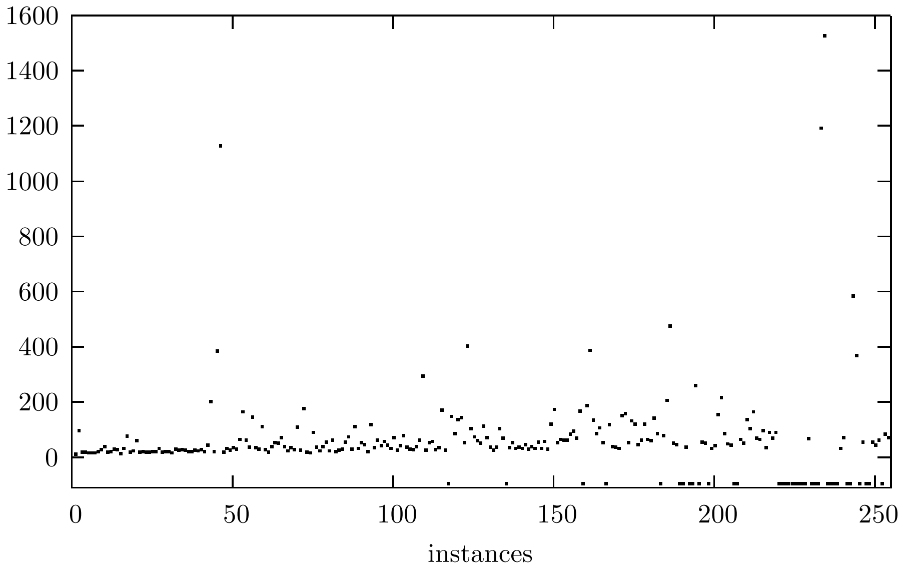


Figure C.31: Total number of LPs

#	name	n	$\|\mathcal{C}\|$	m	best lb	best ub	
117	uni2.1.1-metr_6	477	477	2261	3918200	5491462	mem/flt
135	steinc7.rq.066	500	430	1000	2251	2262	mem/flt
159	WPP-C351	691	189	1236	*11193	11193	mem/flt
166	WPP-C551	718	196	1920	9369	9401	mem/flt
182	vrpfeas8.infty-metr.rq.1	857	542	3403	*11836	opt 11836	time limit
183	h1.n0900.rq.125	900	375	4662	127868	206463	mem/flt
186	WPP-D451	947	248	2161	*11111	opt 11111	time limit
189	WPP-D552	965	252	2643	10864	16880	mem/flt
190	WPP-D371	979	255	1738	*12706	12706	mem/flt
192	WPP-D651	985	256	3036	10631	15817	mem/flt
193	WPP-D652	985	256	3036	10618	17982	mem/flt
194	WPP-D471	996	258	2182	*11506	opt 11506	time limit
195	WPP-D472	996	258	2182	11449	17899	mem/flt
198	WPP-D671	999	259	3073	10940	17648	mem/flt
201	alb1000.c1-r2-rq.15	1000	697	1998	*1915	1915	time limit
202	alb1000.c0-r0-rq.1	1000	807	1998	*2544	opt 2544	time limit
203	steind10.rq.1	1000	807	2000	*4149	opt 4149	time limit
204	steind6.rq.1	1000	807	2000	*4262	opt 4262	2782s
205	steind7.rq.1	1000	807	2000	*4218	opt 4218	2083s
206	steind9.rq.1	1000	807	2000	4198	5587	mem/flt
207	steind10.rq.066	1000	859	2000	*4375	4375	mem/flt
208	steind6.rq.066	1000	859	2000	*4379	opt 4379	2290s
209	steind7.rq.066	1000	859	2000	*4401	opt 4401	2090s
210	steind8.rq.066	1000	859	2000	*4267	opt 4267	time limit
211	steind9.rq.066	1000	859	2000	*4398	opt 4398	1963s
212	steind1.rq.1	1000	878	1250	*6780	opt 6780	1747s
220	steind5.rq.066	1000	913	1250	*7377	7377	mem/flt
221	uni5.1.1-metr_1	1060	1060	5315	27420151	41279940	mem/flt
222	uni5.1.infty-metr_1	1060	1060	12074	19686407	28974319	mem/flt
223	h1.n1100.rq.125	1100	320	6967	*125242	125242	mem/flt
224	uni1.1.1-metr_3	1120	569	7582	4289010	7281269	mem/flt
225	uni1.1.infty-metr_3	1120	569	8350	3120472	5413426	mem/flt
226	uni5.1.1-metr_0	1184	584	6339	14611261	27143193	mem/flt
227	uni5.1.infty-metr_0	1184	584	13024	10553362	20195764	mem/flt
228	h1.n1200.rq.125	1200	568	5306	159554	268746	mem/flt
229	h1.n1300.rq.125	1300	582	6228	156436	272164	time limit
230	h1.n1400.rq.125	1400	588	7224	157231	256477	mem/flt
231	uni1.1.1-metr_4	1416	693	5122	3037739	5515489	mem/flt
232	uni1.1.infty-metr_4	1416	693	10757	2368318	4487088	mem/flt
233	uni3.1.1-metr_4	1432	1432	2838	*15920000	opt 15920000	time limit
234	uni3.1.infty-metr_4	1432	1432	6146	*14810000	opt 14810000	time limit
235	h1.n1500.rq.125	1500	572	8294	161130	289466	mem/flt
236	h1.n1600.rq.125	1600	710	7548	178198	295341	mem/flt
237	uni2.1.1-metr_8	1817	1817	3652	6036004	8713112	mem/flt
238	uni2.1.infty-metr_8	1817	1817	13916	5615486	7944401	mem/flt
239	alb2000.c1-r2-rq.15	2000	1402	3996	*3901	3901	time limit
240	alb2000.c0-r0-rq.1	2000	1630	3996	*5265	5265	time limit
241	uni1.1.1-metr_2	2152	2152	4311	6638484	9847250	mem/flt
242	uni1.1.infty-metr_2	2152	2152	16000	6370613	9894261	mem/flt
243	uni4.1.1-metr_1	2319	2319	4539	*23660000	opt 23660000	time limit
244	uni4.1.infty-metr_1	2319	2319	8830	*23280000	opt 23280000	time limit
245	steine1.rq.1	2500	2204	3125	17005	19324	mem/flt
246	steine2.rq.1	2500	2204	3125	*17862	17862	time limit
247	steine3.rq.1	2500	2204	3125	17903	17904	mem/flt
248	steine4.rq.1	2500	2204	3125	*17624	17624	mem/flt
249	steine5.rq.1	2500	2204	3125	*18097	opt 18097	time limit
250	steine1.rq.066	2500	2286	3125	*17365	17365	time limit
251	steine2.rq.066	2500	2286	3125	*18284	18284	time limit
252	steine3.rq.066	2500	2286	3125	18263	20767	mem/flt
253	steine4.rq.066	2500	2286	3125	17923	17930	time limit
254	steine5.rq.066	2500	2286	3125	*18531	opt 18531	time limit

Table C.1: 61 Challenge instances

List of Figures

List of Tables

List of Algorithms

Bibliography

[ABCC] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding cuts for the TSP. Chapter of a book on the TSP which is currently being prepared.

[ABCC98] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. On the solution of the Traveling Salesman Problem. In *Doc. Math. J. DMV (Extra Volume ICM)*, pages 645–656, 1998.

[ABCC01] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. TSP cuts which do not conform to the template paradigm. In M. Jünger and D. Naddef, editors, *Computational Combinatorial Optimization*, pages 261–303. Springer-Verlag Berlin Heidelberg, 2001.

[ABCC03] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Implementing the Dantzig-Fulkerson-Johnson algorithm for large Traveling Salesman Problems. *Math. Program. Ser. B*, 97(1–2):91–153, 2003.

[AP01] D. Alevras and M. W. Padberg. *Linear optimization and extensions: problems and solutions.* Springer-Verlag Berlin Heidelberg, 2001.

[BCCM85] E. Benavent, V. Campos, A. Corberán, and E. Mota. Analisis de heuristicos para el problema del cartero rural. *Trabajos de Estadistica e Investigacion Operativa*, 36(2):27–38, 1985.

[BCS00] E. Benavent, A. Corberán, and J. M. Sanchis. Linear programming based methods for solving arc routing problems. In M. Dror, editor, *Arc Routing: Theory, Solutions, and Applications*, pages 231–275. Kluwer Academic Publishers, 2000.

[BG86] F. Barahona and M. Grötschel. On the Cycle Polytope of a Binary Matroid. *J. Comb. Theory Ser. B*, 40:40–62, 1986.

[BM76] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* London, Macmillan, 1976.

[Brø83] A. Brøndsted. *An introduction to convex polytopes.* Springer, 1983.

[Car97] R. Carr. Separating clique trees and bipartition inequalities having a fixed number of handles and teeth in polynomial time. *Math. Oper. Res.*, 22(2):257–265, 1997.

[Car04] R. Carr. Separation algorithms for classes of STSP inequalities arising from a new STSP relaxation. *Math. Oper. Res*, 29(1):80–91, 2004.

[Cat88] P. A. Catlin. A reduction method to find spanning Eulerian subgraphs. *J. Graph Theory*, 12:29–45, 1988.

[Cat92] P. A. Catlin. Supereulerian Graphs: A Survey. *J. Graph Theory*, 16(2):177–196, 1992.

[CCCM81] N. Christofides, V. Campos, A. Corberán, and E. Mota. An algorithm for the Rural Postman Problem. Technical report, Imperial College London, 1981.

[CCPS98] W. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* John Wiley & Sons, 1998.

[CFN85] G. Cornuéjols, J. Fonlupt, and D. Naddef. The Traveling Salesman Problem on a Graph and some related Integer Polyhedra. *Math. Program.*, 33:1–27, 1985.

[Chr97] T. Christof. *Low-Dimensional 0/1-Polytopes and Branch-and-Cut in Combinatorial Optimization.* PhD thesis, University of Heidelberg, Germany, 1997.

[Chr98] T. Christof. Porta Forte. Software, 1998.

[CLS01] A. Corberán, A. N. Letchford, and J. M. Sanchis. A cutting plane algorithm for the General Routing Problem. *Math. Program. Ser A*, 90:291–316, 2001.

[Coo] S. Cook. The P versus NP problem. `www.claymath.org/millennium/P_vs_NP/`.

[Cor04] A. Corberán. Personal communication, 2004.

[CR96] T. Christof and G. Reinelt. Combinatorial optimization and small polytopes. *Top*, 4(1):1–53, 1996.

[CR01] T. Christof and G. Reinelt. Decomposition and parallelization techniques for enumerating the facets of combinatorial polytopes. *Int. J. Comput. Geom. Appl.*, 11:423–437, 2001.

[CS94] A. Corberán and J. M. Sanchis. A polyhedral approach to the Rural Postman Problem. *Eur. J. Oper. Res.*, 79:95–114, 1994.

[CS98] A. Corberán and J. M. Sanchis. The General Routing Problem polyhedron: Facets from the RPP and GTSP polyhedra. *Eur. J. Oper. Res.*, 108:538–550, 1998.

[DFJ54] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale Traveling Salesman Problem. *Oper. Res.*, 2:393–410, 1954.

[Die00] R. Diestel. *Graph Theory.* GTM. Springer-Verlag New York, 2000.

[Edm65] J. Edmonds. Maximum matching and a polyhedron with 0-1 vertices. *J. Res. Nat. Bur. Standards*, 69B:125–130, 1965.

[EJ77] J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese Postman Problem. *Math. Program.*, 5:88–124, 1977.

[EL00] R. W. Eglese and A. N. Letchford. Polyhedral theory for arc routing problems. In M. Dror, editor, *Arc Routing: Theory, Solutions, and Applications*, pages 199–230. Kluwer Academic Publishers, 2000.

[Fle99] L. Fleischer. Building Chain and Cactus Representations of All Minimum Cuts from Hao-Orlin in the Same Asymptotic Run Time. *Journal of Algorithms*, 33:51–72, 1999.

[FMGO03] E. Fernández, O. Meza, R. Garfinkel, and M. Ortega. On the Undirected Rural Postman Problem: Tight bounds for based on a new formulation. *Oper. Res.*, 51(2):281–291, 2003.

[Fre79] G. N. Frederickson. Approximation Algorithms for Some Postman Problems. *Journal of the Association for Computing Machinery*, 26(3):538–554, July 1979.

[FU56] G. W. Ford and G. E. Uhlenbeck. Combinatorial problems in the theory of graphs III. In *Proc. Nat. Acad. Sci. USA*, volume 42, pages 529–535, 1956.

[GB93] M. X. Goemans and D. J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Math. Program.*, 60(2):145–166, 1993.

[GH61] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *J. Soc. Ind. Appl. Math.*, 9:551–570, 1961.

[GH87] M. Grötschel and O. Holland. A cutting plane algorithm for minimum perfect 2-matching. *Computing*, 39:327–344, 1987.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[GL00] G. Ghiani and G. Laporte. A branch-and-cut algorithm for the Undirected Rural Postman Problem. *Math. Program. Ser. A*, 87(3):467–481, 2000.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics.* Springer, 2nd edition, 1993.

[GN93] D. Gusfield and D. Naor. Extracting Maximal Information About Sets of Minimum Cuts. *Algorithmica*, 10:64–89, 1993.

[Goe95] M. X. Goemans. Worst-case comparison of valid inequalities for the TSP. *Math. Program.*, 69(2):335–349, 1995.

[GP79] M. Grötschel and M. W. Padberg. On the Symmetric Travelling Salesman Problem I: inequalities. *Math. Program.*, 16:265–280, 1979.

[GR95] Michel X. Goemans and V. S. Ramakrishnan. Minimizing Submodular Functions over Families of Sets. *Combinatorica*, 15(4):499–513, 1995.

[GR00] C. D. Godsil and G. Royle. *Algebraic graph theory.* Springer-Verlag New York, 2000.

[Grü03] B. Grünbaum. *Convex Polytopes.* Springer-Verlag New York, 2nd edition, 2003.

[Gus90] D. Gusfield. Very Simple Methods for all Pairs Network Flow Analysis. *SIAM J. Comput.*, 19(1):143–155, February 1990.

[HLNH99] A. Hertz, G. Laporte, and P. Nanchen-Hugo. Improvement procedures for the Undirected Rural Postman Problem. *INFORMS Journal on Computing*, 11:53–62, 1999.

[HO92] J. Hao and J. B. Orlin. A Faster Algorithm for Finding the Minimum Cut in a Graph. In Greg Frederickson, editor, *Proceedings of the third annual ACM-SIAM Symposium on Discrete Algorithms*, pages 165–174, 1992.

[HO94] J. Hao and J. B. Orlin. A Faster Algorithm for Finding the Minimum Cut in a Directed Graph. *Journal of Algorithms*, 17:424–446, 1994.

[Jan93] K. Jansen. Bounds for the General Capacitated Routing Problem. *Networks*, 23:165–173, 1993.

[JRT00] M. Jünger, G. Rinaldi, and S. Thienel. Practical Performance of Efficient Minimum Cut Algorithms. *Algorithmica*, 26:172–195, 2000. Springer-Verlag New York.

[JT98] M. Jünger and S. Thienel. Introduction to ABACUS—A Branch-And-CUt System. *Oper. Res. Lett.*, 22:83–95, 1998.

[Let96] A. N. Letchford. New inequalities for the General Routing Problem. *Eur. J. Oper. Res.*, 96:317–322, 1996.

[Let97] A. N. Letchford. *Polyhedral results for some Arc Routing problems.* PhD thesis, Lancaster University Management School, 1997.

[Let99] A. N. Letchford. The general routing polyhedron: A unifying framework. *Eur. J. Oper. Res.*, 112:122–133, 1999.

[Let03] A. N. Letchford. Personal communication, 2003.

[Let04] A. N. Letchford. Personal communication, 2004.

[Let05a] A. N. Letchford. Personal communication, 2005.

[Let05b] A. N. Letchford. Personal communication, 2005.

[LR76] J. K. Lenstra and A. H. G. Rinnooy Kan. On general routing problems. *Networks*, 6:273–280, 1976.

[LRT04] A. N. Letchford, G. Reinelt, and D. O. Theis. A faster exact separation algorithm for blossom inequalities. In D. Bienstock and G. Nemhauser, editors, *Integer Programming and Combinatorial Optimization 10*, volume 3064 of *LNCS*, pages 196–205. Springer-Verlag Berlin Heidelberg, 2004.

[Nad02] D. Naddef. Polyhedral Theory and Branch-and-Cut Algorithms for the Symmteric TSP. In Gregory Gutin and Abraham P. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, pages 29–116. Kluwer Academic Publishers, 2002.

[NK94] H. Nagamochi and T. Katmeda. Canonical cactus representation for all minimum cuts. *Japan J. Indus. App. Math.*, 11:343–361, 1994.

[NOI94] H. Nagamochi, T. Ono, and T. Ibaraki. Implementing an efficient minimum capacity cut algorithm. *Math. Program. Ser. A*, 67(3):325–341, 1994.

[NR88] D. Naddef and G. Rinaldi. The Symmetric Traveling Salesman Polytope: New facets from the graphical relaxation. Technical Report R. 248, IASI-CNR Rome, 1988.

[NR91] D. Naddef and G. Rinaldi. The Symmetric Traveling Salesman Polytope and its graphical relaxation: Composition of valid inequalities. *Math. Program.*, 51:359–400, 1991.

[NR93] D. Naddef and G. Rinaldi. The graphical relaxation: A new framework for the Symmetric Traveling Salesman Polytope. *Math. Program.*, 58:53–88, 1993.

[Orl74] C. S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4:35–64, 1974.

[ORT05] M. Oswald, G. Reinelt, and D. O. Theis. Not every GTSP facet induces an STSP facet. In M. Jünger and V. Kaibel, editors, *Integer Programming and Combinatorial Optimization 11*, volume 3509 of *LNCS*, pages 468–482. Springer-Verlag Berlin Heidelberg, 2005.

[PQ80] J.-C. Picard and M. Queyranne. On the structure of all minimum cuts in a network and applications. *Math. Program. Study*, 13:8–16, 1980. North-Holland Publishing Company.

[PR82] M. W. Padberg and M. R. Rao. Odd minimum cut-sets and b-matchings. *Math. Oper. Res.*, 7(1):67–80, 1982.

[PR87] M. W. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Oper. Res. Lett.*, 6:1–7, 1987.

[PR90a] M. Padberg and G. Rinaldi. Facet identification for the Symmetric Traveling Salesman Polytope. *Math. Program.*, 47:219–257, 1990.

[PR90b] M. W. Padberg and G. Rinaldi. An efficient algorithm for the minimum capacity cut problem. *Math. Program.*, 47:19–36, 1990.

[Pul73] W. R. Pulleyblank. *Faces of matching polyhedra.* PhD thesis, University of Waterloo, 1973.

[QW93] M. Queyranne and Y. Wang. Hamiltonian path and Symmetric Travelling Salesman polytopes. *Math. Program.*, 58(1):89–110, 1993.

[Rei91] G. Reinelt. TSPLIB – A Traveling Salesman Problem Library. *ORSA J. Comput.*, 3:376–384, 1991.

[Rei94] G. Reinelt. *The Traveling Salesman — Computational Solutions*, volume 840 of *LNCS*. Springer-Verlag Heidelberg, 1994.

[Riz03] R. Rizzi. A simple minimum T-cut algorithm. *Discrete Appl. Math.*, 129(2–3):539–544, 2003.

[RT06] G. Reinelt and D. O. Theis. A note on the Undirected Rural Postman Problem polytope. To appear in Math. Program. (DOI: 10.1007/s10107-005-0640-1), 2006.

[San90] J. M. Sanchis. *El poliedro del problema del cartero rural.* PhD thesis, University of Valencia, 1990.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

[Sch03] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer-Verlag Berlin Heidelberg, 2003.

[Slo05] N. J. A. Sloane. The on-line encyclopedia of integer sequences. `www.research.att.com/~njas/sequences/`, 2005.

[Wen99] K. M. Wenger. Kaktus-Repräsentation der minimalen Schnitte eines Graphen und Anwendung im Branch-and-Cut Ansatz für das TSP. Master's thesis, University of Heidelberg, Germany, October 1999.

[Wen03] K. M. Wenger. *Generic Cut Generation Methods for Routing Problems.* PhD thesis, University of Heidelberg, 2003.

[Zie98] G. M. Ziegler. *Lectures on Polytopes.* Springer-Verlag New York, 1998.

List of symbols

$[\alpha,\beta]$, $]\alpha,\beta[$, $[\alpha,\beta[$, $]\alpha,\beta]$	intervals in $\mathbb{R}$
(a,α)	inequality $ax \geq \alpha$ (sometimes equation)
$\langle \cdot \mid \cdot \rangle$	inner product
$F \vee G$	join, i.e., smallest upper bound of two elements F and G of a lattice
$x \oplus y$	addition modulo two
$(U:V)$	$(U:V) := \partial(U) \cap \partial(V)$
$\partial(U)$	coboundary of the node set U: edges with precisely one end node in U
$\lvert\cdot\rvert_1$	1-norm of a vector $x \in \mathbb{R}^n$, defined by $\lvert x\rvert_1 := \sum_i \lvert x_i\rvert$
$\mathbf{0}$	all-zeroes vector defined on appropriate index set
$\mathbf{1}$	all-ones vector defined on appropriate index set
$\mathbf{2}^X$	power set of set X, i.e., set of all subsets of X
$\mathbb{\Delta}^k$	k-dimensional standard simplex in $\mathbb{R}^{k+1}$: $\operatorname{conv}\{\chi^j \mid j = 0, \dots, k\}$
χ^F	incidence vector of appropriate length of the set F, i.e., $\chi_i^F = 1$ if $i \in F$ and $\chi_i^F = 0$ otherwise.
χ^j	shorthand for $\chi^{\{j\}}$

$\mathbb{A}^k$	affine space of dimension k: set of all points $x \in \mathbb{R}^{k+1}$ satisfying the equation $\sum_{j=0}^{k} x_j = 1$
$\operatorname{aff} X$	affine hull
$\operatorname{codim} F$	codimension of a face of a polyhedron P $(\dim P - \dim F)$ or pure polyhedral complex C $(\dim C - \dim F)$
$\operatorname{cone} X$	conic hull
$\operatorname{conv} X$	convex hull
$\dim X$	dimension of a set: $\dim X := \dim \operatorname{lin}(X - X)$
$\operatorname{dist}_G(u, v)$	distance between nodes u and v in graph G
E_n	set of all 2-element subsets of $V_n = \{0, \ldots, n-1\}$; edge set of K_n
$E(G)$	edge set of graph G
$E(x)$	set of all edges e of a graph with $x_e \neq 0$
$\mathscr{F}(x)$	set of facets of a polyhedron containing the vertex x
$G[U]$	subgraph induced by node set U
$G \setminus F$	graph with edges in F deleted
$G - U$	graph with nodes in U deleted
$\operatorname{Gr}_X f$	graph of the mapping f defined on X, i.e., $\{(x, f(x)) \mid x \in X\}$
Id	(linear) identity mapping
K_n	complete graph on n nodes, with $V(K_n) = V_n$ and $E(K_n) = E_n$
$\ker f$	kernel of a linear mapping: $\ker f := \{x \mid f(x) = 0\}$
$\operatorname{lin} X$	linear hull
pr_I	projection onto space with index set I, e.g., $\operatorname{pr}_I \colon \mathbb{R}^{I \cup J} \to \mathbb{R}^I$
$\mathbb{Q}$	rational numbers ($\mathbb{Q}_+$ non-negative, $\mathbb{Q}^*$ non-zero, $\mathbb{Q}_+^*$ strictly positive)
$\mathbb{R}$	the reals ($\mathbb{R}_+$ non-negative, $\mathbb{R}^*$ non-zero, $\mathbb{R}_+^*$ strictly positive)
V_n	the set $\{0, \ldots, n-1\}$; node set of K_n
$V(G)$	node set of graph G
$x(F)$	$x(F) := \sum_{f \in F} x_f$
$x\restriction_F$	restriction of function/vector x to set F (see 0.2.2)
x/F	(merged) vector on shrunk graph (see 0.2.2)
$\mathbb{Z}$	the integers ($\mathbb{Z}_+$ non-negative, $\mathbb{Z}^*$ non-zero, $\mathbb{Z}_+^*$ strictly positive)