Universität Trier

New Concise Extended Formulations for Circular Structures in Optimization Problems

DISSERTATION

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

vorgelegt am Fachbereich IV - Mathematik
der Universität Trier.

Author:
Bernd Perscheid, MSc

Berichterstatter:
Prof. Dr. Sven de Vries
Prof. Dr. Eddie Cheng

Eingereicht am: 02. Juli 2020
Disputation am: 09. September 2020

Contents

German summary (Zusammenfassung)

Viele $\mathcal{NP}$-schwere Optimierungsprobleme aus der klassischen Graphentheorie, wie beispielsweise das Stabile-Mengen-Problem oder das Cliquenproblem, wurden in den vergangenen Jahrzehnten ausgiebig erforscht. Häufig besteht die Problemstellung darin, eine optimale Teilmenge der Knoten oder Kanten des zugrunde liegenden Graphen auszuwählen. Zur Lösung werden in der Regel direkte kombinatorische Methoden auf den Graphen angewandt. Eine universelle und sehr einfache Methode ist die vollständige Enumeration. Alle zulässigen Lösungen werden dabei aufgelistet und die beste davon wird schlussendlich ausgewählt. Aufgrund der meist großen Anzahl an zulässigen Lösungen kann dieses Verfahren durch geschicktes Ausschließen bestimmter Suchbereiche während der Ausführung häufig erheblich beschleunigt werden. Alternativ können graphentheoretische Probleme auch als ganzzahlige lineare Programme formuliert werden, deren optimale Lösung dann eine optimale Lösung für das ursprüngliche Problem liefert.

Um ein ganzzahliges lineares Programm zu lösen, lassen sich zunächst die Ganzzahligkeitsbedingungen relaxieren. Anschließend werden Ungleichungen gesucht, welche fraktionale Ecken des für die Relaxierung zulässigen Bereichs, aber keine ganzzahligen Punkte abschneiden. Im Idealfall führt dies zu einer vollständigen Beschreibung der konvexen Hülle aller zulässigen Punkte des ganzzahligen linearen Programms. Leider ist das Finden einer vollständigen Beschreibung selbst für den Fall, dass die entsprechende konvexe Hülle lediglich polynomiell viele Facetten hat, meist viel zu aufwendig. Daraus ergibt sich das Ziel, die schwache Relaxierung des ganzzahligen linearen Programms durch das Hinzufügen starker, für die konvexe Hülle aller ganzzahligen zulässigen Punkte gültiger Ungleichungen bestmöglich zu verstärken.

Viele Klassen gültiger Ungleichungen sind exponentiell groß. Ein einfaches Beispiel hierfür sind die ungeraden Kreisungleichungen des Stabile-Mengen-Problems, da ein Graph exponentiell viele ungerade Kreise enthalten kann. Glücklicherweise lässt sich manchmal in polynomieller Zeit überprüfen, ob ein gegebener Punkt mindestens eine von exponentiell vielen Ungleichungen verletzt. Dies ist beispielsweise für die ungeraden Kreisungleichungen des Stabile-Mengen-Problems der Fall. Ein solches Verfahren wird Separations-Algorithmus genannt. Existiert ein solcher, so existiert auch eine lineare Formulierung, die einen bestimmten Punkt genau dann enthält, wenn dieser keine der entsprechenden Ungleichungen

verletzt.

Diese Arbeit lässt sich inhaltlich in zwei Abschnitte unterteilen, wobei der erste Abschnitt den größten Teil dieser Arbeit bildet. Darin werden insbesondere neue erweiterte Formulierungen für das Stabile-Mengen-Problem, für das allgemeine nichtkonvexe quadratische Programm mit Box-Bedingungen, sowie für das p-Median-Problem vorgestellt. Eine allgemein bekannte erweiterte Formulierung für das Ungerade-Kreis-Polytop des Stabile-Mengen-Problems wird durch eine neue erweiterte Formulierung verbessert. Weiterhin werden verschiedene neue erweiterte Formulierungen für das allgemeine 1-Rad-Polytop des Stabile-Mengen-Problems vorgestellt. Die sogenannten A-ungeraden Kreisungleichungen des nichtkonvexen quadratischen Programms mit Box-Bedingungen können dazu dienen, eine schwache Relaxierung deutlich zu verstärken. Hierfür wird ebenfalls eine erweiterte Formulierung konstruiert und dessen Stärke anhand numerischer Experimente veranschaulicht. Für die Gerichtete-Ungerade-Kreis-Relaxierung des p-Median-Problems sind einige hinreichende Bedingungen bekannt, welche Ganzzahligkeit der Ecken garantieren. Sind diese Bedingungen erfüllt, so lässt sich das p-Median-Problem mit einer neuen erweiterten Formulierung in polynomieller Zeit lösen.

Der zweite Abschnitt dieser Arbeit widmet sich einem bekannten, sehr schnellen Algorithmus zum Finden einer größten Clique in sehr großen, dünn besetzten Graphen. Drei modifizierte Versionen dieses Verfahrens werden hergeleitet und anhand numerischer Experimente mit dem ursprünglichen Algorithmus verglichen. Dabei wird eine deutliche Laufzeit-Verbesserung durch eine dieser neuen Versionen erzielt. Die Stärken der übrigen beiden neuen Versionen kommen mit zunehmender Dichte des zugrunde liegenden Graphen zum Tragen.

Preface

One refereed publication was written, which appears as de Vries et al. (2019) in the bibliography. Besides that, two additional articles are available on `www.optimization-online.org` and appear as de Vries and Perscheid (2019) and de Vries and Perscheid (2020) in the bibliography. My contribution to all of the three articles mentioned above makes up the major part of their content. Therefore, they are integrated into this thesis and their main results are presented in Chapters 4 and 5.

Chapter 1

Introduction

Many $\mathcal{NP}$-hard optimization problems that originate from classical graph theory, such as the maximum stable set problem and the maximum clique problem, have been extensively studied over the past decades and involve the choice of a subset of edges or vertices. There usually exist combinatorial methods that can be applied to solve them directly in the graph. The most simple method is to enumerate feasible solutions and select the best. It is not surprising that this method is very slow oftentimes, so the task is to cleverly discard fruitless search space during the search. An alternative method to solve graph problems is to formulate integer linear programs, such that their solution yields an optimal solution to the original optimization problem in the graph.

In order to solve integer linear programs, one can start with relaxing the integer constraints and then try to find inequalities for cutting off fractional extreme points. In the best case, it would be possible to describe the convex hull of the feasible region of the integer linear program with a set of inequalities. In general, giving a complete description of this convex hull is out of reach, even if it has a polynomial number of facets. Thus, one tries to strengthen the (weak) relaxation of the integer linear program best possible via strong inequalities that are valid for the convex hull of feasible integer points.

Many classes of valid inequalities are of exponential size. For instance, a graph can have exponentially many odd cycles in general and therefore the number of odd cycle inequalities for the maximum stable set problem is exponential. It is sometimes possible to check in polynomial time if some given point violates any of the exponentially many inequalities. This is indeed the case for the odd cycle inequalities for the maximum stable set problem. If a polynomial time separation algorithm is known, there exists a formulation of polynomial size that contains a given point if and only if it does not violate one of the (potentially exponentially many) inequalities.

1.1 Our contribution

This thesis can be divided into two parts. The first part is the main part and it contains various new results. We present new extended formulations for several optimization problems, i.e. the maximum stable set problem, the nonconvex quadratic program with box constraints and the p-median problem. In the second part we modify a very fast algorithm for finding a maximum clique in very large sparse graphs. We suggest and compare three alternative versions of this algorithm to the original version and compare their strengths and weaknesses.

The maximum stable set problem plays a central role in this thesis. We give an overview and a deeper insight into several different relaxations containing specific inequalities in Chapter 2. Some very useful separation algorithms, e.g. for the odd cycle and the 1-wheel inequalities for the stable set polytope, are summarized in Chapter 3. These methods are fundamental for what we present in Chapter 4: a new smaller extended formulation of the odd cycle polytope and two new extended formulations of the 1-wheel polytope.

For approximating the feasible region of the nonconvex quadratic program with box constraints (BoxQP), it can be useful to construct a linear relaxation. Unfortunately, it turns out to be a very weak approximation in general. However, it can be strengthened by the so-called A-odd cycle inequalities. We construct an extended formulation for these inequalities in Chapter 5. Next, we extend relaxations of the nonconvex quadratic program with box constraints for our numerical experiments. Thereby, we confirm the strength of our formulation by showing that much of the optimality gap can be closed with the extended relaxations.

Chapter 6 deals with the p-median problem. We first summarize some remarkable results concerning properties of the underlying digraph that ensure integrality of the directed odd cycle relaxation. Then, we present our new extended formulation that can be used to solve the p-median problem in this case in polynomial time.

Chapters 2–6 focus on linear relaxations for several optimization problems and our main contribution is the construction of extended formulations. In contrast, Chapter 7 deals with enumerative branch-and-bound algorithms for the maximum clique problem. After giving an overview on existing algorithms and strategies, we present three modified versions of a fast algorithm for very large sparse graphs. We demonstrate by our computational experiments that one of them improves the original version. Moreover, we expect that the slower versions that employ vertex colorings are more efficient if the graph is more dense.

1.2 Fundamentals and notation

This section provides an introduction into basic knowledge in graph theory which will be relevant throughout the thesis. Furthermore, we briefly introduce basic terminology to polyhedra and linear programming. The notation we introduce is partially equal or slightly different to the notation that can be found in literature. For a broader overview on fundamentals in graph theory, we refer the reader to Diestel (2010), Grötschel et al. (1988),

Gross and Yellen (2004), Gross and Yellen (2005), Jungnickel (2013), Schrijver (2003), and Tutte (2001). A detailed introduction to linear programming is given for instance by Schrijver (1986). Plenty of theory on polytopes, that are important for linear programming, is provided by Ziegler (2007).

The *maximum stable set problem* and the *maximum clique problem* are classical $\mathcal{NP}$-hard problems, c.f. Garey and Johnson (1979). We consider aspects of both of these graph theoretical problems in this thesis. Therefore, they are defined and an integer linear programming formulation for each of them is given in this section.

1.2.1 Graphs & digraphs

A *graph* is an ordered triple $G = (V, E, \Psi)$. The elements in the set V are called *vertices* and the elements in the set E are called *edges*. The set of unordered pairs of vertices in V, twice the same vertex included, is denoted by $V^{(2)}$. Then, $\Psi : E \to V^{(2)}$ is an incidence function which maps every edge in E to an element in $V^{(2)}$. For each edge $e \in E$ there exist $u, v \in V$ with $\Psi(e) = uv = vu$. The cardinality of V is denoted by n and the cardinality of E is denoted by m. We restrict ourselves to graphs where n and m are finite. Sometimes, e.g. when vertices are denoted as tuples or as numbers, we use the notation $\{u, v\}$ instead of uv. We say that e is *incident* to u and v, or u and v are *incident* to e. If two edges are incident to the same vertex, they are also called *incident*. Furthermore, two vertices u and v are *adjacent* to each other, if there exists an edge being incident to both of them. We call u a *neighbor* of v and v a *neighbor* of u. The set of all neighbors of u is denoted by $N(u)$, i.e. all vertices $w \in V$ with $vw \in E$. An edge e with $\Psi(e) = uu$ is called *loop*. Edges which are incident to the same pair of vertices are said to be *parallel*. In this case, the pair of vertices is connected by *multiple edges*. The *degree* of a vertex v, denoted by $\deg(v)$, is the number of edges that are incident to v, where loops are counted twice. We define

$$\delta(G) \coloneqq \min\{\deg(v) : v \in V\}.$$

For the sake of simplicity and easier reading, graphs without multiple edges are defined by the tuple (V, E). Then, edges can be identified with the vertices they are incident to and the function Ψ can be recovered. Thus, $e = uv = vu$ is an unambiguous assignment. If there exists a graph isomorphic to G, which is embedded into the plane without any edges crossing, we say that G is *planar*.

We define digraphs, where the orientation of the connection between every pair of vertices matters, analogously to graphs. A *digraph* is an ordered triple $D = (V, A, \Psi)$. The elements in the set V are again the *vertices* and the elements in the set A are called *arcs*. The incidence function $\Psi : A \to V \times V$ maps every arc $a \in A$ to an ordered pair of vertices in V. For each arc $a \in A$ there exist $u, v \in V$ with $\Psi(a) = (u, v)$, where u is the *tail* and v is the *head* of a. Then a is said to be *incident* to u and v. Arcs which are incident to the same pair of vertices $u, v \in V$ are said to be *parallel* if either u or v is the tail for each of those arcs. A pair of arcs, that are incident to the same pair of vertices $u, v \in V$, is called *antiparallel* if u is the tail of one of these arcs and the head of the other one. An arc a with

$\Psi(a) = (u, u)$ is a *loop*.

Similarly to graphs, we can clearly determine arcs in a given digraph $D = (V, A)$ without any function Ψ if we do not have parallel arcs. Notice that antiparallel arcs are permitted in this representation of D.

Paths, walks, cycles

A finite sequence $P = (v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ with $k \geq 0$, where vertices and edges are alternating and every edge $e_i \in P$ is incident to v_{i-1} and v_i, is a *walk*. P is a *path* if $v_i \neq v_j$ whenever $i \neq j$. In order to emphasize the start and end point of P, we call the respective walk/path a *v_0-v_k-walk/path*. A walk P with $v_0 = v_k$ is a *closed walk* and if additionally all vertices except v_0 and v_k in P are distinct, P is a *cycle*. Moreover, we call a cycle with k vertices a *k-cycle* and use the notation C_k.

A finite sequence $P = (v_0, a_1, v_1, a_2, v_2, \ldots, a_k, v_k)$ with $k \geq 0$ and alternating vertices and arcs, where v_{i-1} is the tail and v_i is the head for every arc $a_i \in P$, is a *diwalk*. *Dipaths* are defined analogously to paths in the undirected case. We say that a diwalk/dipath P is a *v_0-v_k-diwalk/dipath* if we want to point out that v_0 is the start and v_k is the end point of P. A *closed diwalk* is a diwalk with $v_0 = v_k$. If all vertices except v_0 and v_k in P are distinct, P is called *directed cycle* or short *dicycle*.

In the following chapters we often refer to the number of edges/arcs that are contained in the sequence P. Therefore, a walk/path/cycle is said to be *odd* if the number of (possibly multiply counted) edges is odd. It is called *even* elsewise. A diwalk/dipath/dicycle is said to be *odd*, if the number of (possibly multiply counted) arcs is odd, and *even* otherwise.

If the underlying object is a digraph and this is clear from the context, we omit the prefix "di" for shorter notation. In this case, we consider diwalks/dipaths/dicycles when speaking about walks/paths/cycles.

The number of edges that are contained in P is the *length* of P. Let us denote the edge set of P by E_P. Given some weight function $w : E \to \mathbb{R}$ for a graph $G = (V, E)$, the *weight* of P is a sum of its edge weights and hence

$$w(E_P) \coloneqq \sum_{ij \in E_P} w_{ij}.$$

This definition is applied analogously to the directed case. We use the term *shortest* instead of minimum weight, e.g. a shortest cycle is a minimum weight cycle.

Special classes of graphs and digraphs

We say that a graph G is *simple*, if it has no parallel edges and no loops. G is *connected*, if for every pair $u, v \in V$ with $u \neq v$ there exists a u-v-path.

A graph G is *complete*, if for all $u, v \in V$ with $u \neq v$ the edge uv is contained in E exactly once. D is a *complete* digraph, if the arc set A consists of both arcs (u, v) and (v, u) for every pair $u, v \in V$ with $u \neq v$.

If the vertex set V of G can be partitioned into two sets $V_1 \neq \emptyset$ and $V_2 \neq \emptyset$ such that there does not exist any edge connecting two vertices out of the same set, then G is called *bipartite*.

For a given simple graph $G = (V, E)$, the graph $\bar{G} = (V, \bar{E})$ obtained by having $uv \in \bar{E}$ if and only if $uv \notin E$ is the *complementary graph* of G.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs with $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. Then G_1 is called a *subgraph* of G_2. If $uv \in E_1$ whenever $u, v \in V_1$ and $uv \in E_2$, then G_1 is the *induced subgraph* $G[V_1]$ of G_2 by V_1. If the induced subgraph G_1 of G_2 by V_1 is a cycle, we call this cycle *chordless* in G_2.

A digraph $D = (V, A)$ is called *oriented*, if $(u, v) \in A$ implies $(v, u) \notin A$ for every arc $(u, v) \in A$.

Stable sets, cliques, colorings

When considering stable sets, cliques, and colorings, we always refer to (undirected) graphs throughout this thesis.

A *stable set* $S \subseteq V$ has the property that $uv \notin E$ for every pair $u, v \in S$. S is called *maximal stable set* if there does not exist any vertex $v \in V \backslash S$ such that $S \cup \{v\}$ is a stable set. If S is a stable set and there exists no stable set in G with cardinality greater than $|S|$, then S is a *maximum stable set*. If we are given a weight function $w : V \to \mathbb{R}_+$ for the vertices in G, a stable set with the largest sum of weights among all stable sets in G is a *weighted maximum stable set*. We denote the *incidence vector* of a stable set $S \subseteq V$ by $x^S \in \{0, 1\}^V$, where $x_i^S = 1$ if $i \in S$ and $x_i^S = 0$ otherwise.

A subset $K \subseteq V$ is a *clique* if $uv \in E$ for every pair $u \neq v \in K$. In other words, the induced subgraph of G by K is complete. In order to emphasize that a clique has k vertices, we call it a *k-clique* and use the notation K_k. K is called *maximal clique* if there does not exist any vertex $v \in V \backslash K$ such that $K \cup \{v\}$ is a clique. If K is a clique and there is no clique in G with cardinality greater than $|K|$, then K is a *maximum clique*. If we are given a weight function $w : V \to \mathbb{R}_+$ for the vertices in G, a clique with the largest sum of weights among all cliques in G is a *weighted maximum clique*.

We call an assignment of colors to the vertices of G a *(vertex) coloring* if one color is assigned to each vertex and adjacent vertices are colored differently. Notice that in literature this is often called a *proper vertex coloring*. We consider colorings as a function $c : V \to \mathcal{C}$ with $\mathcal{C} \subseteq \mathbb{N}$ such that $c(u) \neq c(v)$ for all $uv \in E$. Without loss of generality we assume that $\mathcal{C} = \{1, \ldots, k\}$ for some sufficiently large k. If c is surjective, we call c a *k-coloring*. In other words, exactly k different colors appear in a k-coloring in G. Let us denote the set of vertices colored with color i is by V_i. Then $V = V_1 \dot{\cup} \ldots \dot{\cup} V_k$ and each V_i is a stable set in G. Thus, a coloring is a partition of V into disjoint stable sets. A graph is called *k-colorable* if there exists a coloring that uses exactly k colors. Obviously, if G is k-colorable, then G is as well l-colorable for all $l \geq k + 1$. A k-coloring is an *optimal coloring* if and only if G is not $(k-1)$-colorable.

Stability number, clique number, and chromatic number

In this section, the stability number $\alpha(G)$, the clique number $\omega(G)$, and the chromatic number $\chi(G)$ are defined. In the following chapters we occasionally make use of relationships between these numbers in a graph.

Let $w : V \to \mathbb{R}_+$ be any weighting of the vertices in G. Then

$$\alpha(G, w) \coloneqq \max \left\{ \sum_{v \in S} w(v) : S \subseteq V \text{ is a stable set in } G \right\}$$

denotes the maximum weight of a stable set in G. Determining $\alpha(G, w)$ is $\mathcal{NP}$-hard, even in the special case when the weight of all vertices is identical and without loss of generality equal to one, cf. Garey and Johnson (1979). If $w(v) = 1$ for every $v \in V$ and S^* is a maximum stable set, we call

$$\alpha(G) \coloneqq \sum_{v \in S^*} 1 = |S^*|$$

the *stability number* of G, which equals the cardinality of a maximum stable set in G.

Analogously, we define

$$\omega(G, w) \coloneqq \max \left\{ \sum_{v \in K} w(v) : K \subseteq V \text{ is a clique in } G \right\},$$

the maximum weight of a clique in G. If $w(v) = 1$ for every $v \in V$ and K^* is a maximum clique, then

$$\omega(G) \coloneqq \sum_{v \in K^*} 1 = |K^*|$$

is the cardinality of a maximum clique in G. We call $\omega(G)$ the *clique number* of G.

The chromatic number of a graph refers to colorings, which refer to the neighborhood of vertices. Hence, weightings of vertices are unimportant. The chromatic number $\chi(G)$ is defined as

$$\chi(G) \coloneqq \min \{k \in \mathbb{N} : G \text{ is } k\text{-colorable}\}.$$

Notice that every $\chi(G)$-coloring is an optimal coloring in G.

1.2.2 Polyhedra & linear programming

A *polyhedron* is a convex set $P \coloneqq P(A, b) = \{x \in \mathbb{R}^n : Ax \leq b\} \subseteq \mathbb{R}^n$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, i.e. it is the intersection of finitely many closed halfspaces of the form $\{x \in \mathbb{R}^n : \alpha^T x \leq \delta\}$ for some non-zero vector $\alpha \in \mathbb{R}^n$ and some number $\delta \in \mathbb{R}$. The system $Ax \leq b$ is a shorthand for the m many linear inequalities $a_{i,\star} x \leq b_i$ where $i \in \{1, \ldots, m\}$. If a polyhedron P is bounded in the sense that is does not contain a ray $\{x + ty : t \geq 0\}$ for some fixed $x \in P$ and $y \neq \mathbf{0}$, it is called a *polytope*.

An inequality $a_{i,\star} x \leq b_i$ is *redundant* for the system $Ax \leq b$ if it is implied by the other

inequalities in the system. Notice that one redundant inequality can be removed without changing the set of feasible points of $Ax \leq b$. However, it can make other redundant inequalities for the same system irredundant for the reduced system.

Let P be a polyhedron. An inequality $\alpha^T x \leq \delta$ is *valid* for P if it is fulfilled by all points $x \in P$. Any set of the form $F = P \cap \{x \in \mathbb{R}^n : \alpha^T x = \delta\}$ where $\alpha^T x \leq \delta$ is a valid inequality for P is a *face* of P. Every face F of P with $\dim(F) = \dim(P) - 1$ is a *facet* of P. In other words, facets are faces of a polyhedron, except the polyhedron itself, of highest dimension. A vector $x \in P$ is an *extreme point* of P if there does not exist any pair of points $y, z \in P$ with $y \neq x \neq z$ such that $x = \lambda y + (1 - \lambda)z$ for any $\lambda \in [0, 1]$.

For finitely many points $x_1, \ldots, x_k \in \mathbb{R}^n$, $k \in \mathbb{N}$,

$$\operatorname{conv}(\{x_1, \ldots, x_k\}) := \left\{ \lambda_1 x_1 + \ldots + \lambda_k x_k : \lambda_i \geq 0 \; \forall \; i \in \{1, \ldots, k\}, \; \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

is the *convex hull* of the set $\{x_1, \ldots, x_k\}$. Every polytope P is equal to the convex hull of its extreme points. A polytope is called *integral* if it has only integer extreme points.

Linear programming concerns the problem of maximizing or minimizing a linear function over a polyhedron. In an *integer linear program*, we additionally have the constraint that feasible points must have integer components. We sometimes use the shorthand LP and IP, respectively, for linear and integer linear programs. Since these programs are optimization problems, we use the term *(integer) linear optimization problem* simultaneously.

1.2.3 The maximum stable set problem

The *maximum stable set problem* (MSSP) is the problem of finding a weighted maximum stable set in $G = (V, E)$. It can easily be formulated as the integer linear optimization problem

$$\begin{aligned} \max \quad & \sum_{i \in V} w_i x_i & & \\ \text{s.t.} \quad & x_i + x_j \leq 1 & \forall \; ij \in E & \qquad \text{(MSSP)} \\ & x_i \in \{0, 1\} & \forall \; i \in V, & \end{aligned}$$

where $w : V \to \mathbb{R}_+$ is a weighting of the vertices in G. The feasible set of this optimization problem consists of the characteristic vectors of all stable sets in G. Let S be a stable set in G. Two vertices cannot be simultaneously present in S if they are connected by an edge. This is guaranteed by the inequalities $x_i + x_j \leq 1$ for all $ij \in E$. We have $x_i^S = 1$ if $i \in S$ and $x_i^S = 0$ otherwise. We assume $w(v) = 1$ for every $v \in V$ as a special case of the MSSP. The objective function then maximizes over the cardinality of all stable sets in G and hence finds a maximum stable set in G.

1.2.4 The maximum clique problem

The *maximum clique problem* (MCP) is the problem of finding a weighted maximum clique in $G = (V, E)$. Many different equivalent formulations are known, see Bomze et al. (1999). One simple intuitive formulation as an integer linear optimization problem is

$$\begin{aligned} \max \quad & \sum_{i \in V} w_i x_i & & \\ \text{s.t.} \quad & x_i + x_j \leq 1 & \forall\ ij \in \bar{E} & \qquad \text{(MCP)} \\ & x_i \in \{0, 1\} & \forall\ i \in V & \end{aligned}$$

where $w : V \to \mathbb{R}_+$ is a weighting of the vertices in G. This formulation is almost equal to our above formulation of the MSSP. The inequality $x_i + x_j \leq 1$ for all $ij \in \bar{E}$ ensures that if two vertices i and j are not connected by an edge, they cannot be simultaneously present in a clique. Since a stable set in a given graph G is a clique in $\bar{G}$, this is not surprising. As for the MSSP, we assume $w(v) = 1$ for every $v \in V$ as a special case of the MCP and therefore find a maximum clique when solving it.

Chapter 2

Stable sets: inequalities, polytopes, and perfectness

The set of feasible solutions for many every day problems can be expressed as stable sets in a given graph $G = (V, E)$. Let us for example think about the following situation: We are given a social network with n people and we plan to carry out an experiment on how groups of people interact if no one knows each other very well. Every single person in the social network be represented by a vertex in V and two vertices are adjacent if and only if these two people are connected in the social network. Then every potential test group for our experiment is equivalent to a stable set in G. In particular, a largest possible test group can be determined by finding a maximum stable set in G. Another thing we could ask for is, if there exists a subset of k people that qualifies as a candidate for a test group. Or in other words: Is the stability number $\alpha(G)$ at least k?

Besides direct applications of stable sets for modeling and solving practical problems, constraints in many more complex optimization problems can have a structure associated with stable sets. For example, the airline crew scheduling problem with base constraints

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & Ax = \mathbb{1} \\ & d_1 \leq Dx \leq d_2 \\ & x_j \in \{0,1\} \quad \forall\, j = 1, \ldots, n, \end{aligned} \tag{SPB}$$

where $A \in \{0,1\}^{m \times n}$, $d_1, d_2 \in \mathbb{Q}^d$ and $D \in \mathbb{Q}_+^{d \times n}$, is studied by Hoffman and Padberg (1993).

The set packing polytope $\operatorname{conv}\{x \in \mathbb{R}^n : Ax \leq \mathbb{1}, x \in \{0,1\}^n\}$ constitutes a major part of the constraint set of (SPB). Hoffman and Padberg (1993) analyze (SPB) by considering the intersection graph $G_A = (V_A, E_A)$ of matrix A, see Padberg (1973), where $V_A = \{1, \ldots, n\}$

for the column set of A and the set E_A contains an edge uv if and only if $a_{\star,u} + a_{\star,v} \nleq \mathbb{1}$. Every pair of adjacent vertices in G_A represents two columns that cannot be concurrently present in the set packing polytope and vice versa. It follows directly that every valid binary packing must be a stable set.

The task of finding a maximum stable set is $\mathcal{NP}$-hard. Therefore it is unlikely that there exists a formulation of polynomial size for the stable set polytope

$$\mathrm{STAB}(G) \coloneqq \mathrm{conv}\left\{x^S \in \{0,1\}^V : S \subseteq V \text{ is a stable set in } G\right\},$$

the convex hull of the solution space of the MSSP.

In this chapter, we present different types of inequalities which are valid for $\mathrm{STAB}(G)$ and define polytopes $P^X(G)$ arising from the polynomial size edge constrained stable set polytope $P^E(G)$, which has a simple description with polynomially many inequalities. We obtain different polytopes $P^X(G)$ by adding at least one additional class of stable set inequalities to $P^E(G)$. Hence, $\mathrm{STAB}(G) \subseteq P^X(G) \subseteq P^E(G)$. Most of these polytopes $P^X(G)$ are specified by an exponential number of valid inequalities in general. However, the separation problem, cf. Definition 3.1, for some of them can be solved in polynomial time. Furthermore, we recapitulate some definitions and results on the "perfectness" of graphs. We also give examples for graph classes that belong to certain sets of perfect graphs.

Some results require that G has some properties. Parallel edges except one and vertices which are incident to loops are not important for the MSSP. If some of them exist, we could transform G into a simple graph by deleting them. Finding a maximum stable set in an unconnected graph G is equal to finding a maximum stable set in every component of G. Thus, we make the following assumption when considering the MSSP.

Assumption 2.1. *Every graph $G = (V, E)$ we consider is simple and connected. Moreover, $E \neq \emptyset$.*

2.1 Valid inequalities & polytopes

In this section, we present and analyze linear inequalities that are valid for $\mathrm{STAB}(G)$. Since it is out of reach to describe $\mathrm{STAB}(G)$ by a system of linear inequalities in general, we give approximations via different polytopes $P^X(G)$. When speaking about optimization, we always refer to the linear program

$$\begin{aligned} \max \quad & \sum_{i \in V} x_i \\ \text{s.t.} \quad & x \in P^X(G). \end{aligned}$$

2.1.1 The edge constrained stable set polytope

A very simple idea to start with is replacing the integrality constraints $x_i \in \{0,1\}$ for the vertex variables x_i for all $i \in V$ in the MSSP by their linear relaxation. This already yields

two classes of valid inequalities for STAB(G):

$$0 \leq x_i \leq 1 \qquad \forall\, i \in V \qquad \text{(trivial inequalities)} \qquad (2.1')$$

$$x_i + x_j \leq 1 \qquad \forall\, ij \in E \qquad \text{(edge inequalities)} \qquad (2.2)$$

Due to Assumption 2.1, we only consider simple, connected graphs with $E \neq \emptyset$. Therefore, G is not an isolated vertex and it is sufficient to replace (2.1') by

$$x_i \geq 0 \quad \forall\, i \in V \qquad (2.1)$$

since the upper bound 1 for every x_i is implied by the corresponding edge inequality $x_i + x_j \leq 1$ in combination with $x_j \geq 0$ if i is connected to any other vertex $j \in V$.

The edge constrained stable set polytope

$$P^E(G) \coloneqq \{x \in \mathbb{R}^n : x \text{ satisfies (2.1) and (2.2)}\}$$

is the linear relaxation of the solution space of the MSSP and of polynomial size. It consists of just $n + m$ inequalities.

We define

$$\alpha^E(G) \coloneqq \max\left\{\sum_{i \in V} x_i : x \in P^E(G)\right\},$$

the objective value of an optimal solution for $P^E(G)$, the relaxed MSSP.

Remark 2.2. *Let G be a graph. Then $\alpha^E(G) \geq \frac{n}{2}$ holds.*

One can easily check this by considering the vector $x = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)^\mathsf{T} \in \mathbb{R}^n$, which does not violate any inequality describing $P^E(G)$.

We now want to analyze cases in which $P^E(G) = \text{STAB}(G)$. Grötschel et al. (1988) show in Proposition 9.1.3 and their subsequent remark the following property:

Theorem 2.3. *$P^E(G)$ is equal to STAB(G) if and only if G is bipartite.*

Therefore, the edge constrained stable set polytope $P^E(G)$ has extreme points which are infeasible for STAB(G) if G is not bipartite.

Example 2.4. *Let G be a complete graph. Obviously, a maximum stable set is given by a single vertex. Hence, for the stability number $\alpha(G) = 1$ holds. We have seen that $\alpha^E(G) \geq \frac{n}{2}$ by Remark 2.2. This extreme example shows how that the ratio $\alpha^E(G)/\alpha(G)$ can be greater or equal than $\frac{n}{2}$ and thus, that $P^E(G)$ may be a very weak approximation of STAB(G) in general.*

2.1.2 The odd cycle polytope

In Section 2.1.1, we have seen that $P^E(G) = \text{STAB}(G)$ for every bipartite graph. A minimal example graph G for having $P^E(G) \supset \text{STAB}(G)$ is an odd cycle with just three vertices (which is also a clique). More general, no odd cycle can occur in a bipartite graph.

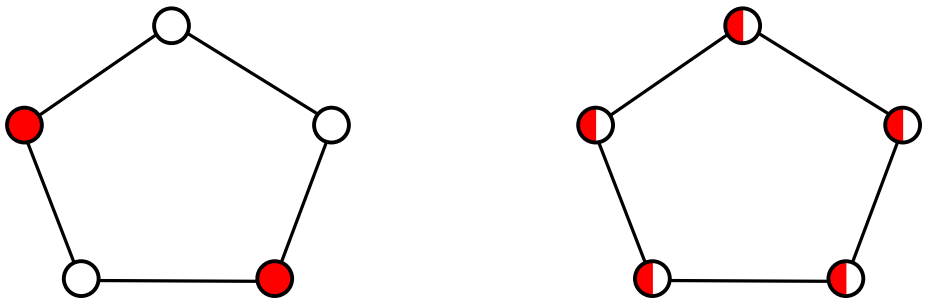
Lemma 2.5. *Every graph with $n \geq 2$ is bipartite if and only if it contains no odd cycle.*

This leads us to another class of valid stable set inequalities. Let G be an odd cycle, for example a 5-cycle as in Figure 2.1(a). One optimal solution for the MSSP, i.e. a stable set with cardinality 2, is highlighted by the red vertices. Figure 2.1(b) shows a fractional extreme point in $P^E(G)$. The objective value of this point is $\frac{5}{2}$, which is greater than 2.

In general, at most half of the vertices in G rounded down can be concurrently present in a stable set for any odd cycle. Thus, inequalities

$$\sum_{i \in C} x_i \leq \frac{|C| - 1}{2} \qquad \forall \text{ odd cycles } C \qquad \text{(odd cycle inequalities)} \tag{2.3}$$

are valid for STAB(G), cf. Grötschel et al. (1988, Chapter 9). Padberg (1973) introduces these inequalities for the odd holes, which are chordless odd cycles and thus a subset of the odd cycles.



(a) An optimal solution for the MSSP.

(b) A fractional extreme point in $P^E(G)$.

Figure 2.1: Stable sets in a 5-cycle.

The odd cycle polytope

$$P^{OC}(G) \coloneqq \{x \in \mathbb{R}^n : x \text{ satisfies (2.1), (2.2), and (2.3)}\},$$

arises from adding the odd cycle inequalities (2.3) to the edge constrained stable set polytope. We illustrate this exemplarily for a 5-cycle in Figure 2.1(b), where $x = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^\mathsf{T} \in P^E(G)$ is excluded from $P^{OC}(G)$ by an odd cycle inequality.

In contrast to $P^E(G)$, the polytope $P^{OC}(G)$ may be given by exponentially many inequalities, since the number of (odd) cycles in a graph is exponential in the number of edges. Arman and Tsaturian (2017) present an exponential upper bound and provide examples of graphs to the reader, where the number of cycles is 1.37^m for sufficiently large m. Although there could be an exponential number of odd cycles in G, one finds an optimal solution for

$P^{OC}(G)$ in polynomial time with an extended formulation. We will present three different extended formulations of $P^{OC}(G)$ in Chapter 4.

We define

$$\alpha^{OC}(G) \coloneqq \max\left\{\sum_{i\in V} x_i : x \in P^{OC}(G)\right\}$$

analogously to $\alpha^{E}(G)$. Because every odd cycle includes at least 3 vertices, a trivial lower bound on $\alpha^{OC}(G)$ can be explicitly given.

Remark 2.6. *For any graph G, we have $x = \left(\frac{1}{3},\dots,\frac{1}{3}\right)^{\mathsf{T}} \in P^{OC}(G)$. Hence, $\alpha^{OC}(G) \geq \frac{n}{3}$ holds.*

2.1.3 The clique polytope

Since a clique in a graph is a complete subgraph and the stability number of a complete graph is 1, a stable set can contain at most one vertex of a clique. Consequently, another exponential class of valid inequalities for the stable set polytope, which was introduced by Padberg (1973), is given by the clique inequalities:

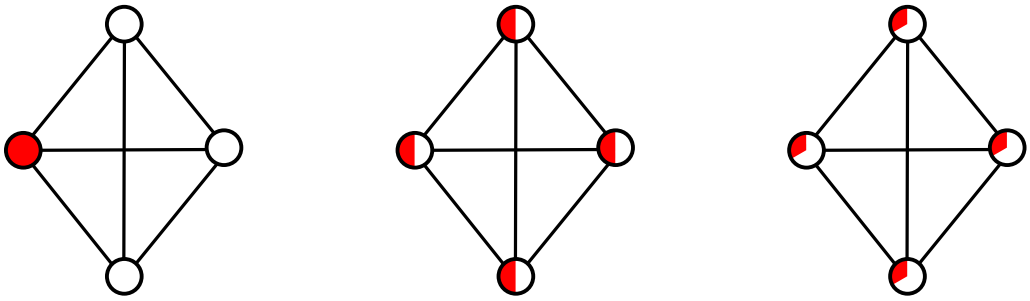
$$\sum_{i\in K} x_i \leq 1 \qquad \forall \text{ cliques } K \qquad \text{(clique inequalities)} \tag{2.4}$$

The corresponding clique polytope is defined as

$$P^{K}(G) \coloneqq \{x \in \mathbb{R}^n : x \text{ satisfies (2.1) and (2.4)}\}.$$

Obviously, every edge in G is a 2-clique. This implies that the edge inequalities (2.2) are a subset of the clique inequalities (2.4). Therefore, we just have to add inequalities for cliques of size at least 3 to $P^{E}(G)$.

Let G be a 4-clique. One of four optimal solutions for the MSSP is depicted in Figure 2.2(a). Again, we consider the vector $x \in \mathbb{R}^n$ with $x_i = \frac{1}{2}$ for all $i \in V$, which is feasible for $P^{E}(G)$, see Figure 2.2(b). This vector yields a total value of 2 with respect to the objective function of the MSSP. Now, one can argue that this is not a feasible point for the odd cycle polytope in Section 2.1.2 as it is cut off by every single odd cycle inequality from the four 3-cycles that appear as subgraphs in G. The vector $x = \left(\frac{1}{3},\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)^{\mathsf{T}}$, illustrated in Figure 2.2(c), is feasible for $P^{OC}(G)$. It yields an objective value of $\frac{4}{3}$, which is larger than 1. In general, there are points in $P^{OC}(G)$ that violate the clique inequalities for cliques K with $|K| \geq 4$.



(a) An optimal solution for the MSSP.

(b) A fractional extreme point in $P^E(G)$.

(c) A fractional extreme point in $P^{OC}(G)$.

Figure 2.2: Stable sets in a 4-clique.

We define

$$\alpha^K(G) \coloneqq \max\left\{\sum_{i\in V} x_i : x \in P^K(G)\right\}$$

as the optimal objective value of the optimization problem over the clique polytope. In contrast to the edge constrained stable set polytope and the odd cycle polytope, we do not have any fixed vector for an arbitrary graph G, which satisfies all clique inequalities *and* might provide a higher objective value than $\alpha(G)$.

Remark 2.7. *Let n be an arbitrary, fixed number. There is no better lower bound than 1 for $\alpha^K(G)$ which is valid for all graphs G with n vertices, since $\sum_{i\in V} x_i \leq 1$ is a valid clique inequality for the complete graph.*

Next, we take a closer look at the number of clique inequalities that appear in the description of $P^K(G)$. Unfortunately, there may exist exponentially many cliques in a graph. Anyway, we can hope that we do not require an inequality for each clique in G.

Example 2.8. *Let G be a complete graph. Then every nonempty subset of V is a clique. Although we have $2^n - 1$ inequalities in this case, every clique inequality $\sum_{i\in K} x_i \leq 1$ for $K \subset V$ is dominated by $\sum_{i\in V} x_i \leq 1$, because $x_i \geq 0$ for all $i \in V$ are valid inequalities for $P^K(G)$. Therefore, $\sum_{i\in V} x_i \leq 1$ is the only facet inducing clique inequality of $P^K(G)$.*

A more general and even stronger result was proven by Padberg:

Theorem 2.9. *(Padberg, 1973) Let K be a clique. Then the inequality $\sum_{i\in K} x_i \leq 1$ defines a facet of* STAB(G) *if and only if K is a maximal clique in G.*

This leads to the questions of how many maximal cliques can exist in a graph and if the number of maximal cliques is polynomially bounded from above. Moon and Moser (1965) have shown, that every graph has at most $3^{n/3}$ maximal cliques and that this bound is attained by some graph.

One of the best algorithms in practice to list all maximal cliques was introduced by Bron and Kerbosch (1973). Fortunately, there exist several graph classes, where we are able to list all maximal cliques in polynomial time. These graph classes, such as complete graphs, triangle-free graphs, and chordal graphs, have only *a few* maximal cliques which can be listed efficiently, cf. Rosgen and Stewart (2007). In their recent work *"A linear time algorithm for maximal clique enumeration in large sparse graphs"*, Yu and Liu (2017) present an algorithm to enumerate all maximal cliques in graphs with special properties in linear time. Significant parameters for the execution time have to be small and this is the case if the given graph is large and sparse enough.

For general graphs, Grötschel et al. (1981) show that it is not possible to optimize over $P^K(G)$ in polynomial time. Hence, we are not able to derive an extended formulation for $P^K(G)$ with polynomially many variables and inequalities as for $P^{OC}(G)$.

Theorem 2.10. *(Grötschel et al., 1988, Theorem 9.2.9) The optimization problem for $P^K(G)$ is $\mathcal{NP}$-hard.*

Let us compare the odd cycle inequalities from the previous section to the clique inequalities. At first glance, the clique inequalities seem to be much stronger in general. In fact, this is not true. For example, if the largest clique in G is a 2-clique, then $x = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)^\mathsf{T}$ is feasible for $P^K(G)$. Consider the 5-cycle in Figure 2.1(b), where an odd cylce inequality is required to ensure that there exists an integer optimal solution for the relaxation of the MSSP, such as the solution in Figure 2.1(a). If G has many odd cycles and no clique of size greater or equal than 3, an optimal point in $P^K(G)$ will provide a much higher objective value than an optimal point in $P^{OC}(G)$. Conversely, we have seen by the example in Figure 2.2(c) that one clique inequality might be better than a few odd cycle inequalities.

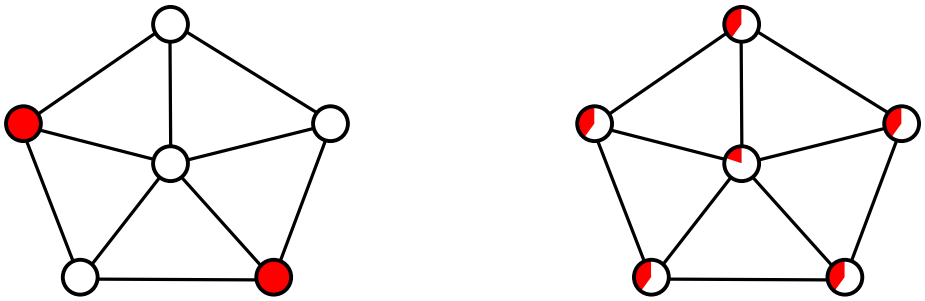
In the next section we will see that in general there exist $x \in P^{OC}(G) \cap P^K(G)$ with $x \notin \mathrm{STAB}(G)$.

2.1.4 The 1-wheel polytope

An important class of valid stable set inequalities was introduced by Cheng and Cunningham (1997). Before we present and explain the different types of *1-wheel inequalities*, let us consider a small, simple example for the sake of motivation.

Figure 2.3(a) shows a graph where one (of five) optimal integral solution is highlighted. We can certify its optimality by reasoning that if the vertex in the middle belongs to a stable set S, it must be the only one and if any other vertex belongs S, there is a cycle inequality that only allows for one more vertex to belong to S.

The fractional solution $x = (x_1, x_2, x_3, x_4, x_5, x_6)^\mathsf{T} = \left(\frac{1}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)^\mathsf{T}$, where x_1 is the variable for the vertex with degree 5, is shown in Figure 2.3(b). Here, the objective value is $\sum_{i \in V} x_i = \frac{11}{5}$, which is larger than 2. We can see in the three figures below that no odd cycle inequality and no clique inequality is violated by x. Notice that we have five different 3-cycles as in Figure 2.4(a), which are as well cliques, five different 5-cycles as in Figure 2.4(c) and the 5-cycle in Figure 2.4(b) in this 1-wheel. Each of the corresponding odd cycle inequalities, except the inequalities for odd cycles as given in Figure 2.4(c), is fulfilled



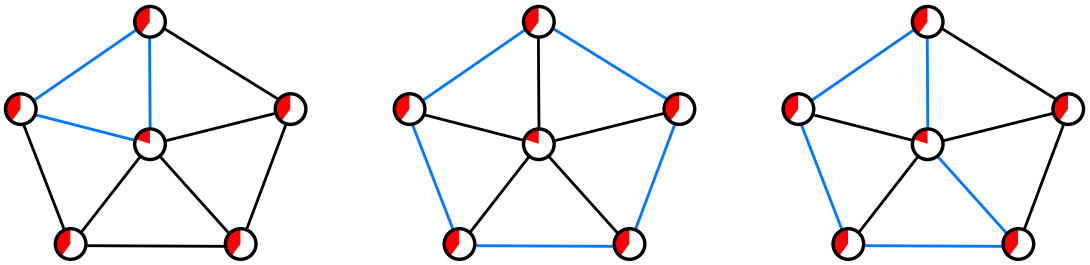
(a) An optimal solution for the MSSP.

(b) A fractional extreme point in $P^{OC}(G) \cap P^{K}(G)$, whose coordinates are in $\{\frac{1}{5}, \frac{2}{5}\}$.

Figure 2.3: Stable sets in a 1-wheel.

with equality. The slack of the respective inequality for every odd 5-cycle that includes x_1 is $\frac{1}{5}$.

We can generalize this example to any odd cycle with an additional vertex which is adjacent to every vertex of the odd cycle.



(a) (b) (c)

Figure 2.4: Odd cycle and clique inequalities that are not violated.

Let C be an odd cycle, $h \notin C$, and every vertex of C be connected to h by an edge. Then the inequality

$$\frac{|C|-1}{2} x_h + \sum_{i \in C} x_i \leq \frac{|C|-1}{2} \tag{2.5}$$

is valid for the stable set polytope. If $x_h = 1$ or the odd cycle inequality for C is satisfied with equality, there is no slack for the corresponding inequality (2.5).

Inequalities (2.5) are just a small subclass of the 1-wheel inequalities. Therefore, we define the class of 1-wheels, Cheng and Cunningham (1997) have introduced. We slightly deviate from their notation and use an equivalent definition to keep our notation consistent throughout this thesis. Moreover, we recapitulate their results with regard to valid stable set inequalities for these 1-wheels.

Definition 2.11. *(de Vries et al., 2019, Definition 3.1) A graph W with vertices $V_W = C \dot\cup S \dot\cup R \dot\cup \{h\}$ is called* simple 1-wheel *if W can be constructed in the following way:*

1. *$C = \{v_1, \dots, v_{2k+1}\}$ is the vertex set of an odd cycle;*
2. *the edges $v_i h$ are added for all $i = 1, \dots, 2k+1$;*
3. *some of the edges $v_i h$ are (multiply) subdivided and S is constituted by the inner vertices of the subdivisions of these edges;*
4. *some of the the edges $v_i v_{i+1}$ are (multiply) subdivided and R is constituted by the inner vertices of the subdivisions of these edges;*
5. *every cycle $h - \dots - v_i - \dots - v_{i+1} - \dots - h$ of W, which does not include any vertex $v \in C \setminus \{v_i, v_{i+1}\}$, is odd.*

We use the convention $v_{2k+2} = v_1$ and call the vertex h the hub *and the elements in C the* spoke ends.

An example is given in Figure 2.5, where $C = \{v_1, v_2, v_3, v_4, v_5\}$, $S = \{s_1, s_2, s_3, s_4, s_5\}$ and $R = \{r_1, r_2, r_3, r_4\}$.

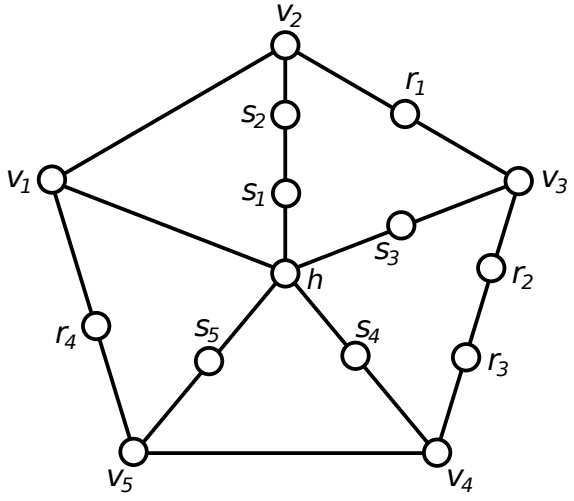


Figure 2.5: A simple 1-wheel.

Definition 2.12. *Let W be a simple 1-wheel. For every $i \in \{1, \dots, 2k+1\}$, we denote the spoke path connecting the hub h to v_i via zero or more vertices in S by P_{h,v_i}. Analogously, the rim path connecting the spoke end v_i to v_{i+1} via zero or more vertices in R is denoted by $P_{v_i,v_{i+1}}$.*

The vertices in C are partitioned into

$$\mathcal{E} \coloneqq \{v_i \in C : P_{h,v_i} \text{ is of even length (in the number of edges)}\},$$

which are called even spoke ends*, and*

$$\mathcal{O} \coloneqq \{v_i \in C : P_{h,v_i} \text{ is of odd length (in the number of edges)}\},$$

which are called odd spoke ends*.*

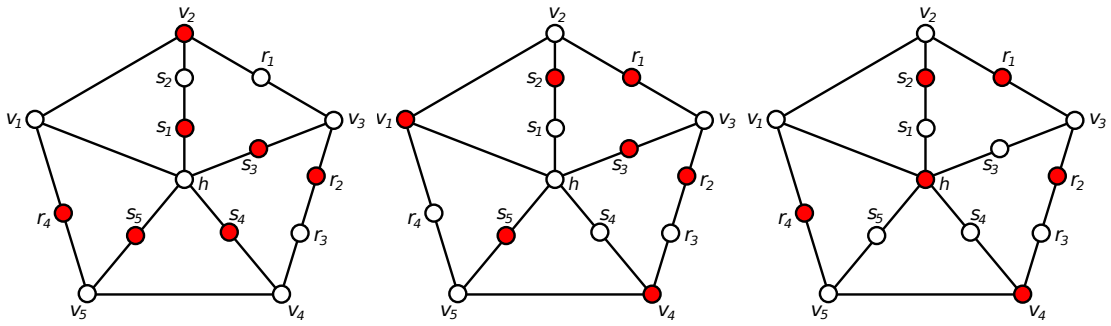
Every vertex of a simple 1-wheel belongs to exactly one of the sets $\mathcal{E}$, $\mathcal{O}$, S, R, $\{h\}$. In the example in Figure 2.5, the odd cycle $C = \{v_1, v_2, v_3, v_4, v_5\}$ is partitioned into $\mathcal{E} = \{v_3, v_4, v_5\}$ and $\mathcal{O} = \{v_1, v_2\}$.

Cheng and Cunningham (1997) show that the 1-wheel inequalities

$$kx_h + \sum_{i=1}^{2k+1} x_{v_i} + \sum_{v \in \mathcal{E}} x_v + \sum_{v \in S \cup R} x_v \leq k + \frac{|S| + |R| + |\mathcal{E}|}{2} \tag{$I_\mathcal{A}$}$$

$$(k+1)x_h + \sum_{i=1}^{2k+1} x_{v_i} + \sum_{v \in \mathcal{O}} x_v + \sum_{v \in S \cup R} x_v \leq k + \frac{|S| + |R| + |\mathcal{O}| + 1}{2} \tag{$I_\mathcal{B}$}$$

for simple (and, as we will see, nonsimple) 1-wheels are valid for STAB(G) and they give sufficient conditions for them to be facet-inducing for STAB(G).



(a) Stable set S_1. (b) Stable set S_2. (c) Stable set S_3.

Figure 2.6: Stable sets in a 1-wheel.

Example 2.13. *Consider the simple 1-wheel in Figure 2.5. The right hand side of both inequalities $(I_\mathcal{A})$ and $(I_\mathcal{B})$ is 8. A maximum stable set in this simple 1-wheel has cardinality*

7. *The stable sets S_1 in Figure 2.6(a) and S_2 in Figure 2.6(b) are maximum stable sets, whereas the stable set S_3 in Figure 2.6(c) is maximal, but not a maximum stable set. For the incidence vectors of S_2 and S_3, inequalities $(I_{\mathcal{A}})$ and $(I_{\mathcal{B}})$ are fulfilled with equality. The incidence vector of the maximum stable set S_1 leaves a slack of 1 to inequality $(I_{\mathcal{A}})$ and fulfills $(I_{\mathcal{B}})$ with equality.*

The simple 1-wheels defined in Definition 2.11 are themselves a subset of a much larger class of graphs:

Definition 2.14. *A* nonsimple 1-wheel *is a graph constructed by identifying (zero or more) pairs of nonadjacent vertices of a simple 1-wheel.*

Identifying vertices may yield parallel edges. They are irrelevant for stable set problems and hence every parallel edge but one can be removed. Notice that simple 1-wheels can be considered to be a special case of nonsimple 1-wheels.

In contrast to simple 1-wheels, $\mathcal{E}$, $\mathcal{O}$, S and R are *multisets* for nonsimple 1-wheels and one or more pairs of nonadjacent vertices can belong to more than one of these multisets. After the identification of vertices and addition of the respective coefficients, valid inequalities $(I_{\mathcal{A}})$ and $(I_{\mathcal{B}})$ arise for STAB(G).

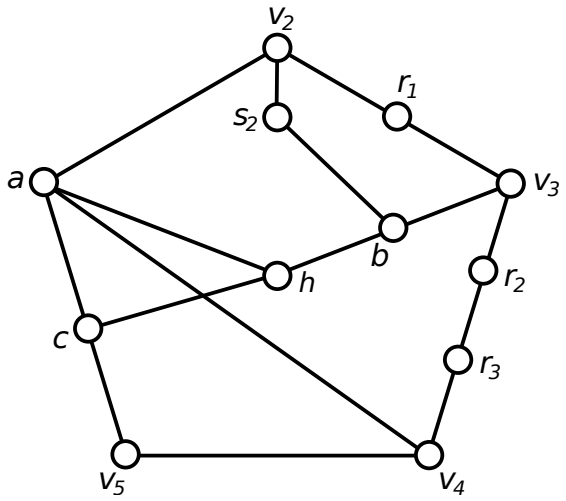


Figure 2.7: A nonsimple 1-wheel.

Example 2.15. *The nonsimple 1-wheel in Figure 2.7 is obtained by the identification of the pairs $\{v_1, s_4\}$, $\{s_1, s_3\}$, and $\{r_4, s_5\}$ of the simple 1-wheel in Figure 2.5. The odd closed walk a, c, v_5, c, h, a originates from the odd face cycle $v_1, r_4, v_5, s_5, h, v_1$.*

Since simple 1-wheels are a special case of nonsimple 1-wheels, where each vertex can only belong to just one of the sets $\mathcal{E}$, $\mathcal{O}$, S and R, we use the general term *1-wheel* in what follows.

With regard to inequalities $(I_{\mathcal{A}})$, the variable x_v for $v \in \mathcal{E}$ appears twice on the left hand side. The same holds for the variable x_v for $v \in \mathcal{O}$ in inequalities $(I_{\mathcal{B}})$. Therefore, we would have to be aware if vertex v belongs to $\mathcal{E}$ or $\mathcal{O}$. Fortunately, there is a very useful result from de Vries (2015), which makes things become easier.

Lemma 2.16. *(de Vries, 2015, Lemma 3 and 4) Every $(I_{\mathcal{A}})$-inequality is representable by an $(I'_{\mathcal{A}})$-inequality. Every $(I_{\mathcal{B}})$-inequality on a 1-wheel without odd spokes of length 1 is representable by an $(I'_{\mathcal{B}})$-inequality.*

Notice that $\mathcal{E} = \emptyset$ for the $(I'_{\mathcal{A}})$-inequalities implies that every spoke end belongs to $\mathcal{O}$. For the $(I'_{\mathcal{B}})$-inequalities, the set $\mathcal{O}$ is empty and therefore every spoke end is in $\mathcal{E}$. Thus, we call the inequalities

$$kx_h + \sum_{i=1}^{2k+1} x_{v_i} + \sum_{v \in S \cup R} x_v \leq k + \frac{|S| + |R|}{2} \tag{$I'_{\mathcal{A}}$}$$

odd 1-wheel inequalities and the inequalities

$$(k+1)x_h + \sum_{i=1}^{2k+1} x_{v_i} + \sum_{v \in S \cup R} x_v \leq k + \frac{|S| + |R| + 1}{2} \tag{$I'_{\mathcal{B}}$}$$

even 1-wheel inequalities. Every 1-wheel inequality in G (except the $(I_{\mathcal{B}})$-inequalities for 1-wheels where at least one spoke is a single edge) is satisfied if no odd and no even 1-wheel inequality is violated due to Lemma 2.16. Hence, we will restrict ourselves to satisfying $(I'_{\mathcal{A}})$ and $(I'_{\mathcal{B}})$.

We define the odd 1-wheel polytope as

$$\begin{aligned} P^{W_{\mathcal{A}}}(G) &:= P^{OC}(G) \cap \left\{x \in \mathbb{R}^n : x \text{ satisfies every inequality } (I'_{\mathcal{A}})\right\} \\ &= \left\{x \in \mathbb{R}^n : x \text{ satisfies (2.1), (2.2), (2.3), and every } (I'_{\mathcal{A}})\right\} \end{aligned}$$

and the even 1-wheel polytope as

$$\begin{aligned} P^{W_{\mathcal{B}}}(G) &:= P^{OC}(G) \cap \left\{x \in \mathbb{R}^n : x \text{ satisfies every inequality } (I'_{\mathcal{B}})\right\} \\ &= \left\{x \in \mathbb{R}^n : x \text{ satisfies (2.1), (2.2), (2.3), and every } (I'_{\mathcal{B}})\right\}. \end{aligned}$$

The 1-wheel polytope

$$P^{W}(G) := P^{W_{\mathcal{A}}}(G) \cap P^{W_{\mathcal{B}}}(G) \subseteq P^{OC}(G)$$

is constituted by the inequalities that describe $P^{OC}(G)$ and additionally all odd and even 1-wheel inequalities. We denote the objective value of an optimal solution when optimizing

over the 1-wheel polytope by

$$\alpha^W(G) \coloneqq \max\left\{\sum_{i\in V} x_i : x \in P^W(G)\right\},$$

similarly to the previously defined alphas.

Lemma 2.17. *The vector* $x = \left(\frac{1}{4},\dots,\frac{1}{4}\right)^\mathsf{T}$ *is in* $P^W(G)$ *for any graph* G*. Thus,* $\alpha^W(G) \geq \frac{n}{4}$ *holds.*

Proof. Remark 2.6 implies that $x = \left(\frac{1}{4},\dots,\frac{1}{4}\right)^\mathsf{T} \in P^{OC}(G)$. Therefore, our task is to prove that no odd or even 1-wheel inequality is violated. Substituting this vector into $(I'_{\mathcal{A}})$ yields

$$\begin{aligned}
&\frac{1}{4}k + \frac{1}{4}(2k+1) + \frac{1}{4}(|S\cup R|) \leq k + \frac{|S|+|R|}{2} \\
\iff\quad &\frac{3}{4}k + \frac{1}{4} \leq k + \frac{2(|S|+|R|) - |S\cup R|}{4} \\
\iff\quad &\frac{1}{4} \leq \frac{1}{4}k + \frac{2(|S|+|R|) - |S\cup R|}{4}.
\end{aligned}$$

If we substitute x into $(I'_{\mathcal{B}})$, we get

$$\begin{aligned}
&\frac{1}{4}(k+1) + \frac{1}{4}(2k+1) + \frac{1}{4}(|S\cup R|) \leq k + \frac{|S|+|R|+1}{2} \\
\iff\quad &\frac{3}{4}k + \frac{1}{2} \leq k + \frac{2(|S|+|R|) - |S\cup R| + 2}{4} \\
\iff\quad &\frac{1}{2} \leq \frac{1}{4}k + \frac{2(|S|+|R|) - |S\cup R| + 2}{4} \\
\iff\quad &0 \leq \frac{1}{4}k + \frac{2(|S|+|R|) - |S\cup R|}{4}.
\end{aligned}$$

Both inequalities are fulfilled for every $k \geq 1$ and arbitrary S and R. Hence, $x \in P^W(G)$ and the validity $\alpha^W(G) \geq \frac{n}{4}$ follows immediately. □

Remark 2.18. *A 4-clique is a (simple) 1-wheel where* $k = 1$ *and* $S = R = \emptyset$*. Consequently,* $\alpha^W(G) = \alpha^K(G) = 1 = \frac{n}{4}$ *if* G *is a 4-clique. This shows that the lower bound on* $\alpha^W(G)$ *in Lemma 2.17 is tight.*

2.1.5 More classes of inequalities

The classes of inequalities presented in the previous sections are central to this thesis. Nevertheless, there exist more classes that have been studied in the last decades. Therefore, we mention some of them that also have been of interest and that can define facets of STAB(G). However, we forego defining polytopes that include some of these inequalities, as we do not investigate separation algorithms or extended formulations for them.

Padberg (1973) and Nemhauser and Trotter (1974) consider odd holes in G. An odd cycle C is an odd hole if and only if the subgraph induced by the vertices of C remains an odd cycle.

Definition 2.19. *(Padberg, 1973; Nemhauser and Trotter, 1974) Let C be an odd hole. Then its complement $\bar{C}$ is called* odd antihole.

We generalize this definition and call the complement of an odd cycle also an odd antihole. It is easy to see that the inequalities

$$\sum_{i \in \bar{C}} x_i \leq 2 \qquad \forall \text{ odd antiholes } \bar{C} \qquad \text{(odd antihole inequalities)} \tag{2.6}$$

are valid stable set inequalities. If one vertex of $\bar{C}$ belongs to a stable set, there are only two more candidates in $\bar{C}$ left. As these two vertices share an edge, only one of them can be present in this stable set.

The web inequalities and the antiweb inequalities are also valid for STAB(G). They are introduced and analyzed by Trotter (1975). Webs and antiwebs are defined as follows.

Definition 2.20. *Let $p \geq 2$ and $1 \leq q \leq \frac{p}{2}$. Then the graph $W(p,q)$ with vertex set $V_{W(p,q)} = \{1, \ldots, p\}$ and edge set*

$$E_{W(p,q)} = \left\{ij : j \in \{i+q, \ldots, i-q \pmod n\} \ \forall \ i \in V_{W(p,q)}\right\}$$

is called web.

The modulo function in the definition of the edge set may produce the vertex sequence $k, \ldots, l$ with $k > l$. In this context we define it as the sequence $k, \ldots, p, 1, \ldots, l$, similar to the notation of cycles.

Definition 2.21. *Let $W(p,q)$ be a web. The complement $\bar{W}(p,q)$ is called* antiweb.

The web inequalities are defined as

$$\sum_{i \in V_{W(p,q)}} x_i \leq \quad q \qquad \forall \text{ webs } W(p,q) \qquad \text{(web inequalities)} \tag{2.7}$$

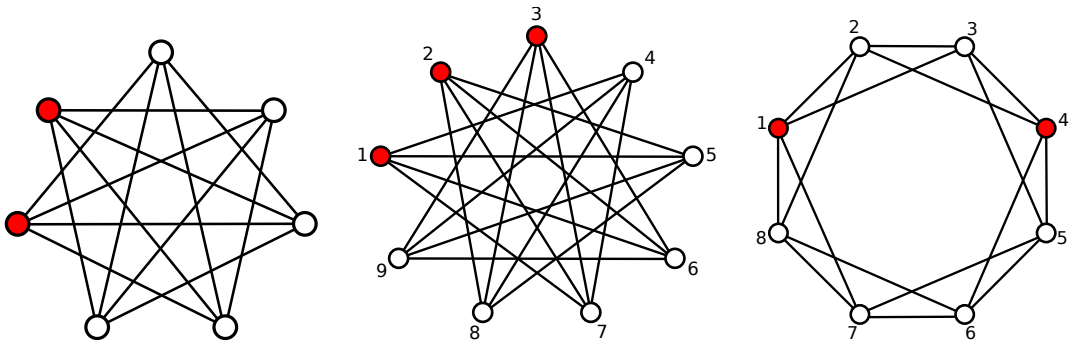
and generalize other inequalities from the previous sections. A web $W(p,1)$ for example is a p-clique, the web $W(k+1,k)$ is a k-cycle and a web $W(k+1,2)$ is an odd antihole with k vertices for any integer $k \geq 2$, cf. Coniglio and Gualandi (2014).

Moreover,

$$\sum_{i \in V_{\bar{W}(p,q)}} x_i \leq \left\lfloor \frac{p}{q} \right\rfloor \qquad \forall \text{ antiwebs } \bar{W}(p,q) \qquad \text{(antiweb inequalities)} \tag{2.8}$$

are as well valid stable set inequalities.

For each of the three graph classes we consider in this section, we present an example in Figure 2.8. These graph classes yield further valid inequalities for STAB(G). A maximum stable set for each graph in Figure 2.8 is given by the red vertices.



(a) Odd antihole $\bar{C}$ ($|\bar{C}| = 7$). (b) Web $W(9, 3)$. (c) Antiweb $\bar{W}(8, 3)$.

Figure 2.8: Maximum stable sets in odd antiholes, webs, and antiwebs.

A very general class of inequalities for stable sets in G are the *rank inequalities* that were introduced by Chvátal (1975):

$$\sum_{i \in U} x_i \leq \alpha(G[U]) \qquad \forall\ U \subseteq V \qquad \text{(rank inequalities)} \tag{2.9}$$

They bound the number of vertices in a subset $U \subseteq V$ that can be concurrently present in a stable set S in G by the stability number of the induced subgraph of G by U.

The rank inequalities are a very large class of inequalities as one can imagine that induced subgraphs of G can be graphs like odd cycles, cliques, 1-wheels, webs or anything else. For an extensive analysis we recommend the work of Coniglio and Gualandi (2014), who present the first formulation for the separation problem of these inequalities as a bilevel integer program.

2.2 Perfectness of graphs

It may be useful to classify graphs that allow solving the MSSP in polynomial time. In this section, we show that specific polytopes $P^X(G)$ are equal to the stable set polytope STAB(G) for some given graph classes. We recapitulate some results on classes that have been studied extensively in the past. Moreover, we define two new classes.

Definition 2.22. *(Grötschel et al., 1988, Chapter 9) A graph G is called*

a) t-perfect, *if* $\text{STAB}(G) = P^{OC}(G)$,

b) perfect, *if* $\text{STAB}(G) = P^{K}(G)$,

c) h-perfect, *if* $\mathrm{STAB}(G) = P^{OC}(G) \cap P^{K}(G)$.

We will see in Chapter 3 (and hence in Chapter 4) that the MSSP for t-perfect graphs can be solved in polynomial time. For example, the following graphs are t-perfect, cf. Grötschel et al. (1988):

- Bipartite graphs.
- Almost bipartite graphs.
- Series-parallel graphs.
- Nearly bipartite planar graphs.

A sufficient condition for a graph to be t-perfect was proven by Gerards and Schrijver (1986):

Theorem 2.23. *(Gerards and Schrijver, 1986, Corollary 2) If a graph G does not contain a subdivision of K_4 as a subgraph, such that each of the four cycles that include exactly three vertices with degree 3 is odd, then G is t-perfect.*

Remark 2.24. *The excluded subgraphs of Theorem 2.23, that cannot occur in t-perfect graphs, are just simple 1-wheels with $|C| = 3$.*

The maximal clique enumeration algorithm of Yu and Liu (2017) can be used to describe $P^{K}(G)$ with polynomially many linear constraints if the underlying graph is sufficiently large and sparse. Hence, we obtain that the MSSP is polynomially solvable for perfect graphs if they fulfill both properties. By using orthonormal representation constraints, see for example Grötschel et al. (1988), one can show that the MSSP can be solved in polynomial time for every perfect graph, even if the graph is not large and sparse.

Theorem 2.25. *The MSSP for perfect graphs is solvable in polynomial time.*

This follows directly by Corollary 9.3.32 of Grötschel et al. (1988), who also give, amongst others, the following examples for perfect graphs:

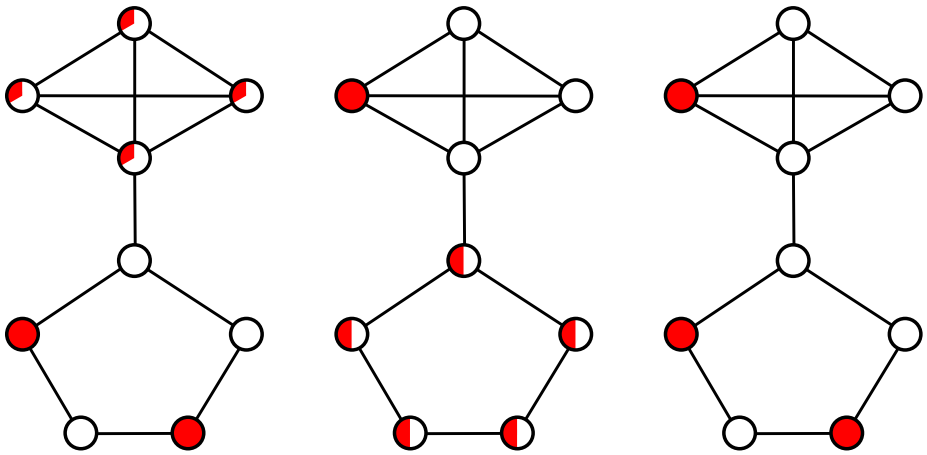
- Bipartite graphs.
- Line graphs of bipartite graphs.
- Interval graphs.
- Triangulated graphs.
- The complements of the above mentioned graphs.

For a detailed analysis of these and some more classes of perfect graphs, we refer the reader to Golumbic (2004).

We now present a simple example for an h-perfect graph that is neither t-perfect nor perfect. For this purpose, we use the following very useful observation.

Remark 2.26. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two vertex and edge disjoint graphs. Furthermore, let S_1 and S_2 be maximum stable sets in G_1 and G_2, respectively. If $S_1 \dot\cup S_2$ is a stable set in $H = (V_1 \dot\cup V_2, E_1 \dot\cup E_2 \dot\cup E_3)$ where E_3 is a set of additional edges, then $S_1 \dot\cup S_2$ is a maximum stable set in H.*

Example 2.27. *Consider the graph G arising from connecting a 4-clique and a 5-cycle by a single edge. Obviously, $\alpha(G) = 3$, see Remark 2.26. This graph is neither t-perfect nor perfect. The vector $x = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 1, 0, 0, 1\right)^\mathsf{T}$ for example, see Figure 2.9(a), is in the odd cycle polytope. It is not in $\mathrm{STAB}(G)$, as the objective value for the maximum stable set problem is $\frac{10}{3} > 3$. On the other hand, $y = \left(0, 1, 0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^\mathsf{T}$ is in the clique polytope, see Figure 2.9(b). The objective value for the maximum stable set problem provided by y is $\frac{7}{2} > 3$ and this implies $P^K(G) \supset \mathrm{STAB}(G)$. For verification that G is h-perfect, we computed the minimal $\mathcal{V}$-representation of $P^{OC}(G) \cap P^K(G)$ with the Parma Polyhedra Library of Python, cf. Bagnara et al. (2016) and showed that all vertices of the polytope are integral. Since all variables are bounded between 0 and 1, we get $P^{OC}(G) \cap P^K(G) = \mathrm{STAB}(G)$. An extreme point z with an objective value of 3 for the maximum stable set problem is presented in Figure 2.9(c).*



(a) $x \in P^{OC}(G)$ (b) $y \in P^K(G)$ (c) $z \in P^{OC}(G) \cap P^K(G)$

Figure 2.9: An h-perfect graph.

Since the 1-wheels as described in Section 2.1.4 do not belong to the class of h-perfect graphs in general, we introduce two new graph classes. Remember that $P^W(G) \subseteq P^{OC}(G)$ by the definition of the 1-wheel polytope.

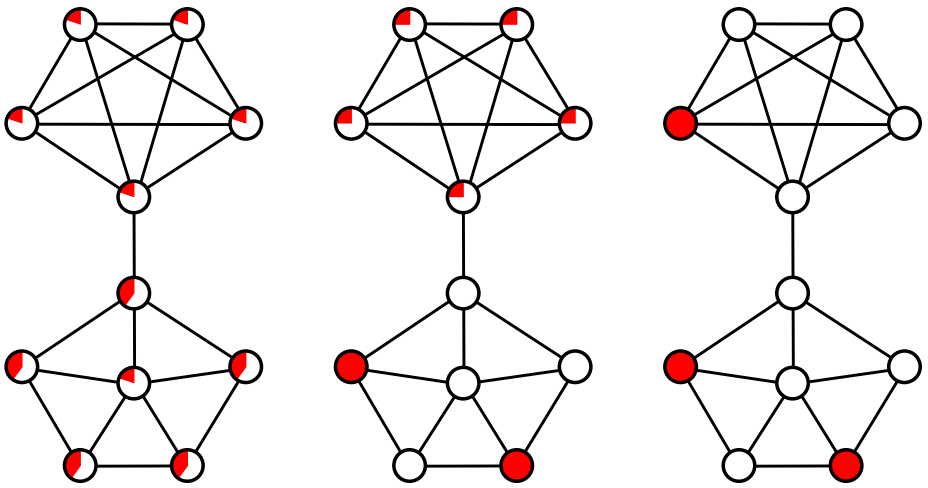
Definition 2.28. *A graph G is called*

a) w-perfect, *if* $\mathrm{STAB}(G) = P^W(G)$,

b) hw-perfect, *if* $\text{STAB}(G) = P^W(G) \cap P^K(G)$.

This definition implies that 1-wheels are w-perfect. Due to Lemma 2.17, $\alpha^W(G) \geq \frac{k}{4} > 1$ for $k > 4$. Thus, a k-clique with $k > 4$ cannot be w-perfect. We now want to give an example of a graph which is neither h-perfect nor w-perfect, but hw-perfect.

Example 2.29. *Let G be the graph that arises from connecting a simple 1-wheel with $|C| = 5$, $S = \emptyset$, and $R = \emptyset$ to a 5-clique by an edge. Notice that $\alpha(G) = 3$ by Remark 2.26, since at most two vertices of this simple 1-wheel and at most one vertex of this clique can be concurrently present in the same stable set, and a stable set with cardinality 3 is presented in Figure 2.10(c). The graph G is not h-perfect, because $x = \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{2}{5}, \frac{2}{5}, \frac{1}{5}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}\right)^\mathsf{T} \in P^{OC}(G) \cap P^K(G)$ with an objective value of $\frac{16}{5} > 3$ for the maximum stable set problem, see Figure 2.10(a). G is not w-perfect, since we allow $x_i = \frac{1}{4}$ for all i in the 5-clique in absence of the clique inequalities. The vector $y = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, 1, 0, 0, 0, 1\right)^\mathsf{T} \in P^W(G)$ provides an objective value of $\frac{13}{4} > 3$ for the maximum stable set problem, see Figure 2.10(b).*



(a) $x \in P^{OC}(G) \cap P^K(G)$ (b) $y \in P^W(G)$ (c) $z \in P^W(G) \cap P^K(G)$

Figure 2.10: An *hw*-perfect graph.

Chapter 3

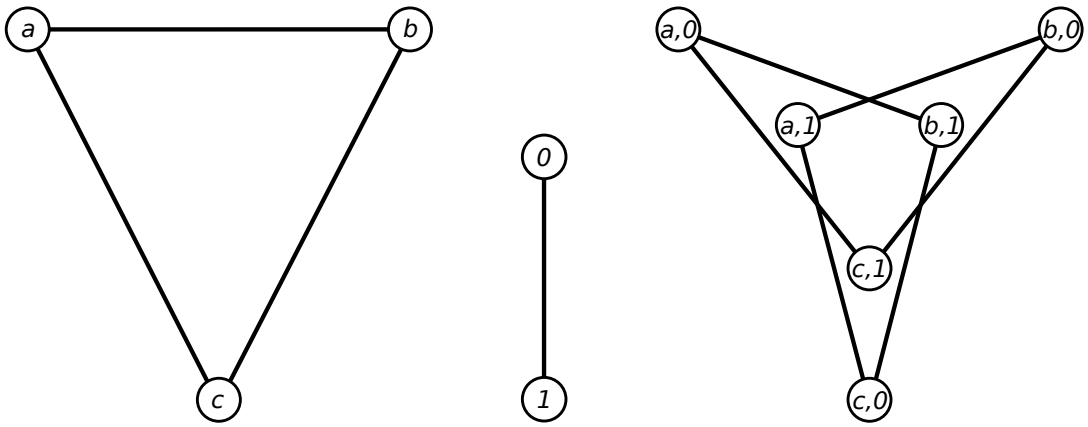
Separation algorithms for stable set relaxations

This chapter covers separation algorithms for the odd cycle inequalities and the 1-wheel inequalities of the stable set polytope, which form the basis for the extended formulations we present in Chapter 4. We first show how shortest path problems can be solved with linear programs, as the separation algorithms we present are based on the weight of shortest paths in specific auxiliary graphs. Next, we explain how the separation problem for the odd cycle inequalities can be solved in polynomial time. Thereafter, two different polynomial time separation algorithms for the 1-wheel inequalities are presented. The separation algorithm of de Vries (2015) is preferable if the underlying graph G is sparse or has a medium density. The separation algorithm of Cheng and Cunningham (1997) is competitive if G is very dense. Finally, we provide an overview of separation algorithms and heuristics for further stable set inequalities.

Definition 3.1. *Given two polytopes P^X and P with $P^X \subseteq P$, the separation problem for P^X and a given vector $\bar{x} \in P$ is to give a valid inequality of P^X that is violated by $\bar{x}$ if one exists.*

Grötschel et al. (1981) show that there exists a polynomial time algorithm for a given linear optimization problem if and only if the associated separation problem can be solved in polynomial time. We will use this general relation between separation and optimization for the construction of extended formulations that are presented in Chapter 4.

In many stable set relaxations, the violation of an inequality by a given $\bar{x}$ is equivalent to some conditions to the underlying graph G or to some auxiliary graph arising from G. For this purpose, we define products of graphs/digraphs, which can be a very useful class of auxiliary graphs. Hammack et al. (2011) define, analyze, and use different types of graph products, which are mainly restricted to the undirected case. We require the so-called *categorical graph product* and extend the definition given by them to digraphs and to combinations of graphs and digraphs.



(a) G (b) H (c) $G \times H$

Figure 3.1: The categorical product of an undirected 3-cycle and an edge.

Definition 3.2. *The* categorical product $G \times H$ *of*

a) two graphs $G = (V_G, E_G)$ *and* $H = (V_H, E_H)$ *is given by the vertex set* $V_{G\times H} = V_G \times V_H$ *and the edge set* $E_{G\times H} = \big\{\{(u,i),(v,j)\} \,:\, uv \in E_G \text{ and } ij \in E_H\big\}$,

b) a graph $G = (V_G, E_G)$ *and a digraph* $F = (V_F, A_F)$ *is given by the vertex set* $V_{G\times F} = V_G \times V_F$ *and the arc set* $A_{G\times F} = \big\{((u,i),(v,j)) \,:\, uv \in E_G \text{ and } (i,j) \in A_F\big\}$,

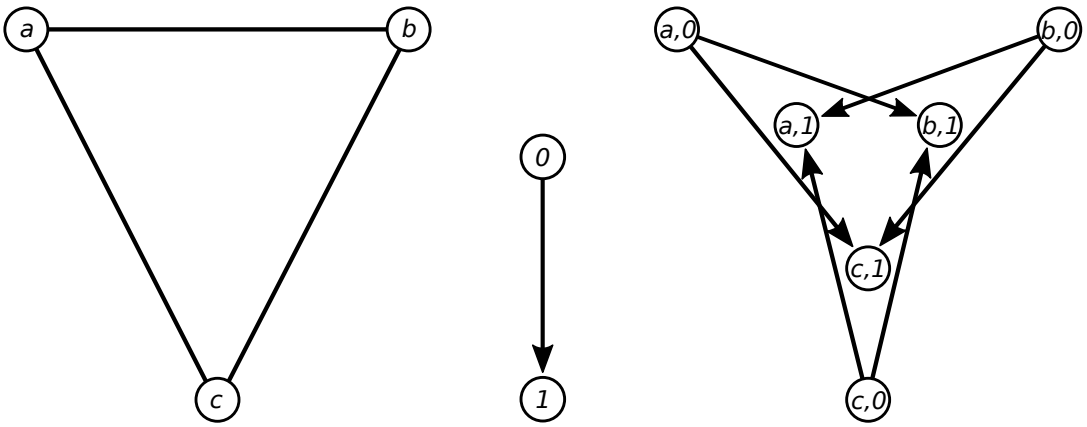
c) two digraphs $D = (V_D, A_D)$ *and* $F = (V_F, A_F)$ *is given by the vertex set* $V_{D\times F} = V_D \times V_F$ *and the arc set* $A_{D\times F} = \big\{((u,i),(v,j)) \,:\, (u,v) \in A_D \text{ and } (i,j) \in A_F\big\}$.

Remember, we refer to edges when using the notation ij (or $\{i,j\}$ if necessary) and to arcs when writing (i,j). A vertex in the categorical product of graphs/digraphs contains two arguments. Consequently, $\{(u,i),(v,j)\}$ denotes an edge between vertices (u,i) and (v,j), whereas $((u,i),(v,j))$ is an arc with tail (u,i) and head (v,j).

An example of the categorical product of two graphs as defined in Definition 3.2a) is given in Figure 3.1, where G is a cycle with 3 vertices and H is a single edge, i.e. $G \times H = (V_{G\times H}, E_{G\times H})$ with $V_{G\times H} = \big\{(a,0),(a,1),(b,0),(b,1),(c,0),(c,1)\big\}$ and

$$E_{G\times H} = \big\{\{(a,0),(b,1)\},\{(a,0),(c,1)\},\{(a,1),(b,0)\},\{(a,1),(c,0)\},\\ \{(b,0),(c,1)\},\{(b,1),(c,0)\}\big\}.$$

Figure 3.2 shows an example of the categorical product of a graph and a digraph, see Definition 3.2b), where G is again a cycle with 3 vertices as in the previous example and F is a single arc.



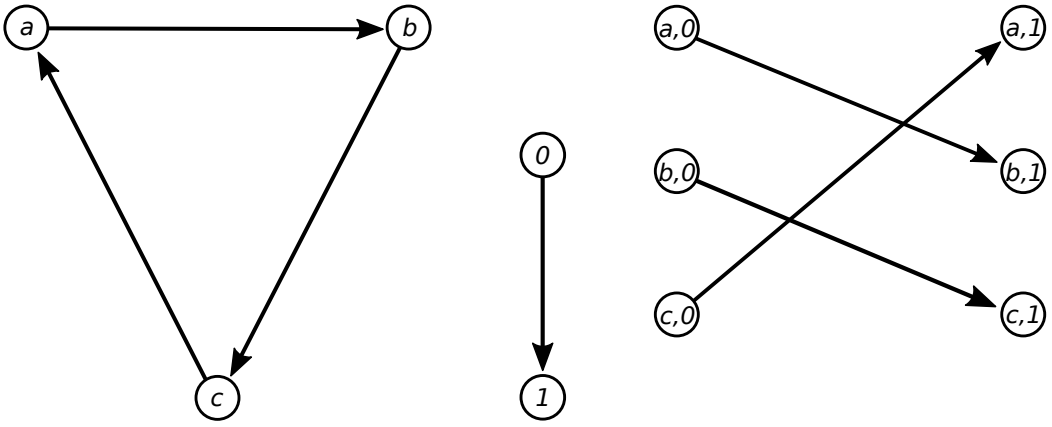
(a) G (b) F (c) $G \times F$

Figure 3.2: The categorical product of an undirected 3-cycle and an arc.

The vertex set of $G \times F$ is $V_{G\times F} = \big\{(a,0),(a,1),(b,0),(b,1),(c,0),(c,1)\big\}$ and its arc set is

$$A_{G\times F} = \big\{((a,0),(b,1)),((a,0),(c,1)),((b,0),(a,1)),((b,0),(c,1)),\\ ((c,0),(a,1)),((c,0),(b,1))\big\}.$$

In Figure 3.3, we illustrate the categorical product of a graph and a digraph as defined in Definition 3.2c), where D is dicycle with 3 vertices and F is a single arc.



(a) D (b) F (c) $D \times F$

Figure 3.3: The categorical product of a directed 3-cycle and an arc.

In this example, we have $V_{D\times F} = \big\{(a,0),(a,1),(b,0),(b,1),(c,0),(c,1)\big\}$ and

$$A_{D\times F} = \big\{((a,0),(b,1)),((b,0),(c,1)),((c,0),(a,1))\big\}.$$

The separation algorithms we present in what follows all have a similar structure. First, an auxiliary graph with certain edge or arc weights is constructed. Next, the weight of a shortest path is computed, whose weight has to be compared to given values to solve the corresponding separation problem. Remember, we do not refer to minimum length but to minimum weight when using the term *shortest* for paths, walks or cycles.

3.1 Solving shortest path problems

In this section, we give an overview on shortest path algorithms that can be applied directly to graphs/digraphs and show how shortest path problems can be solved via linear programming methods.

3.1.1 Graph algorithms

Depending on the structure of the underlying graph/digraph, the algorithm with the best running time should be chosen for finding shortest paths. We collect some efficient algorithms that can be found for example in Even (2012), Jungnickel (2013), and Schrijver (2003).

Let $G = (V, E)$ be a graph whose edges have uniform weights, w.l.o.g. $w_{ij} = 1$ for every $ij \in E$, and let $v \in V$ be fixed. Then the algorithm of Moore finds shortest paths from v to each vertex in G in $\mathcal{O}(n + m)$ time.

For a given digraph $D = (V, A)$ with $w_{ij} \geq 0$ for every $(i, j) \in A$, Dijkstra's algorithm in its original form finds shortest paths from one start vertex $v \in V$ to every vertex in D in $\mathcal{O}(n^2)$. Notice that this running time can be improved to $\mathcal{O}(n \log n + m)$ by using Fibonacci heaps. If we are interested in shortest paths between all pairs of vertices, we could run this algorithm n times, which consumes $\mathcal{O}(n^2 \log n + nm)$ runtime.

In contrast to Dijkstra's algorithm, the Bellman-Ford method can be applied to digraphs with possibly negative arc weights if there does not exist any cycle that has negative total weight. It has complexity $\mathcal{O}(nm)$ if we want to compute the distances from a fixed vertex $v \in V$ to each vertex in D. If we ask for shortest paths between every pair of vertices in D, we could run the Bellman-Ford method n times. This would consume $\mathcal{O}(n^2 m)$ runtime.

In the case that the underlying digraph is not extremely sparse, that is we do not have $m < n$, we would prefer the Floyd-Warshall algorithm which has complexity $\mathcal{O}(n^3)$.

For a broad overview on shortest path algorithms we refer the reader to Schrijver (2003). They also compare running times and explain how the algorithms mentioned above work.

3.1.2 Linear programming

We show how shortest path problems on a digraphs $D = (V, A)$ without negative weight cycles can be solved with linear programs. Both versions, which are related to each other by duality, are presented by Ahuja et al. (1993). If we are given a graph $G = (V, E)$ instead of a digraph and all edge weights are nonnegative, we can solve shortest path problems in G by replacing every edge $uv \in E$ by two antiparallel arcs (u, v) and (v, u). If G has at least one edge with negative weight, this would produce at least one negative weight cycle in the constructed digraph.

Let s and t be fixed vertices in V. We are able to find a shortest s-t-path by solving the minimum cost flow problem

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} w_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{j:\ (i,j)\in A} x_{ij} - \sum_{j:\ (j,i)\in A} x_{ji} = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & i \in V \setminus \{s,t\} \end{cases} \\ & x_{ij} \geq 0 \qquad \forall\ (i,j) \in A. \end{aligned} \tag{MCF-SP}$$

This linear program can be interpreted as sending one unit of flow from s to t through the network. The variables x_{ij} represent the traffic on arc (i, j) and obviously have to be nonnegative. The other constraints ensure that one unit leaves the source s, one unit arrives at the sink t and that the flow conservation property is fulfilled. A solution of this minimum cost flow problem then gives the weight of a shortest s-t-path.

Using a dual relation proposed by Ahuja et al. (1993), the optimal value of the linear program

$$\begin{aligned} \max \quad & y_{st} \\ \text{s.t.} \quad & y_{ss} = 0 \\ & y_{sj} \leq y_{si} + w_{ij} \qquad \forall\ (i,j) \in A \end{aligned} \tag{DMCF-SP}$$

also gives the weight of a shortest s-t-path. Notice that strictly speaking the objective is to maximize $y_{st} - y_{ss}$ where $y_{ss} = 0$ due to the first constraint. In the case that no s-t-path exists, y_{st} is unbounded.

The following system of (in)equalities arises from (DMCF-SP) and has feasible points if and only if the weight of all s-t-paths is at least c:

$$\begin{aligned} & y_{ss} = 0 \\ & y_{sj} \leq y_{si} + w_{ij} \qquad \forall\ (i,j) \in A \\ & y_{st} \geq c \end{aligned} \tag{F-SP}$$

3.2 Separation of the odd cycle inequalities

In this section, we explain how to separate the potentially exponentially many odd cycle inequalities, cf. Section 2.1.2, of the stable set polytope in polynomial time. This method is described by Grötschel et al. (1988) in their proof of Lemma 9.1.11. We recapitulate their idea while using a more detailed description.

For the separation of the odd cycle inequalities, we assume $\bar{x} \in [0,1]^n$ fulfills the edge inequalities (2.2), that is $\bar{x} \in P^E(G)$, which can be easily verified by substituting $\bar{x}$ into all $2n+m$ inequalities. We define weights $\bar{w}_{ij} \coloneqq 1 - \bar{x}_i - \bar{x}_j$ for every edge $ij \in E$, which are obviously nonnegative since $\bar{x} \in P^E(G)$ and hence $\bar{x}_i + \bar{x}_j \leq 1$. Let C be an odd cycle in G. The corresponding odd cycle inequality

$$\sum_{i \in C} \bar{x}_i \leq \frac{|C|-1}{2}$$

of $P^{OC}(G)$ is equivalent to

$$1 \leq |C| - 2\sum_{i \in C} \bar{x}_i = \sum_{ij \in E_C} (1 - \bar{x}_i - \bar{x}_j) = \sum_{ij \in E_C} \bar{w}_{ij} =: \bar{w}(E_C).$$

This leads to the following statement.

Corollary 3.3. *Let $\bar{x} \in P^E(G)$. All odd cycle inequalities are fulfilled by $\bar{x}$ if and only if every odd cycle C has weight $\bar{w}(E_C)$ at least 1.*

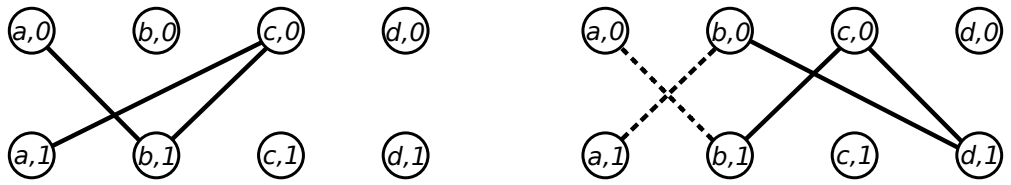
Therefore, computing a shortest odd cycle in G with respect to edge weights $\bar{w}$ suffices to solve the separation problem for the odd cycle inequalities. The first polynomial time algorithm on this problem was given by Grötschel and Pulleyblank (1981). They find a shortest odd cycle containing $ij \in E$ by computing a shortest even path between i and j. Applying this procedure to every edge $ij \in E$, they compute m odd cycles and choose the shortest among these. We use an alternative approach and describe how to find a shortest odd cycle in G as presented in Grötschel et al. (1988, Chapter 8.3). Besides that, we interpret their construction through the concept of categorical graph products.

Consider the categorical product of $G = (V_G, E_G)$ and $H = (V_H, E_H)$ with $V_H = \{0,1\}$ and $E_H = \big\{\{0,1\}\big\}$. An example is given in Figure 3.1, where G is a 3-cycle.

For every edge ij in G, we assign the weight $\bar{w}_{ij}$ to the edges $\{(i,0),(j,1)\}$ and $\{(i,1),(j,0)\}$ in $G \times H$. Since all edge weights $\bar{w}$ are nonnegative, shortest paths in $G \times H$ can be computed efficiently, for instance with the algorithm of Dijkstra (1959). Then the weight of a shortest odd cycle in G can be found in the following way: Compute a shortest path between vertices $(i,0)$ and $(i,1)$ in $G \times H$ for every $i \in V_G$, then choose (one of) the shortest among these paths. The weight of this path is equal to the weight of a shortest odd cycle in G. Checking whether this weight is at least 1 solves the separation problem for the odd cycle inequalities.

A shortest path between a pair $(i,0)$ and $(i,1)$ of vertices in $G \times H$ corresponds to an odd closed walk in G, which can be an odd *cycle*, e.g. the 3-cycle (a,b,c,a) in G arises from the

$(a, 0)$-$(a, 1)$-path in Figure 3.4(a). In general, this odd closed walk in G needs not necessarily be itself an odd cycle in G, even if we consider a shortest of all shortest $(i, 0)$-$(i, 1)$-paths for $i \in V_G$.



(a) (b)

Figure 3.4: Shortest $(a, 0)$-$(a, 1)$-paths in $G \times H$.

For example, the $(a, 0)$-$(a, 1)$-path in Figure 3.4(b) yields the odd closed walk (a, b, c, d, b, a) in G, which is not a cycle. In our example, this is a $(b, 0)$-$(b, 1)$-path, which corresponds to the odd cycle (b, c, d, b) in G. In general, if a shortest $(i, 0)$-$(i, 1)$-path does not give an odd cycle in G, there exists a subpath between $(u, 0)$ and $(u, 1)$ in $G \times H$ for some $u \in V$, whose weight is equal to the weight of a shortest $(i, 0)$-$(i, 1)$-path and which gives an odd cycle in G. It cannot have less weight, since this would contradict the minimality of the weight of the $(i, 0)$-$(i, 1)$-path. Moreover, the nonnegative edge weights ensure that its weight cannot exceed the weight of the $(i, 0)$-$(i, 1)$-path.

3.3 Separation of the 1-wheel inequalities

A polynomial time separation algorithm for a small subclass of the 1-wheel inequalities, which are defined in Section 2.1.4, is presented by Grötschel et al. (1988). This subclass includes all simple 1-wheels where $S = R = \emptyset$, cf. Definition 2.11. Cheng and Cunningham (1997) extend this result to separate *all* 1-wheel inequalities. A few years later, de Vries (2015) presents a faster separation algorithm. He describes how to check if $\bar{x} \in P^{W_A}(G)$ and if $\bar{x} \in P^{W_B}(G)$ in polynomial time. This procedure involves computing the weights of shortest walks in auxiliary graphs. Using a categorical graph product, he improves the $O(n^4)$ running time of the separation algorithm for 1-wheel inequalities by Cheng and Cunningham (1997) and achieves an overall running time of $O(n^2 m + n^3 \log n)$. In contrast to the algorithm of Cheng and Cunningham, there is a dependence on the density of the graph because of the parameter m instead of the n^2-term.

3.3.1 The algorithm of de Vries

For the separation of the 1-wheel inequalities, we require $\bar{x} \in P^{OC}(G)$ and the weights of shortest odd and even walks between pairs of vertices $u, v \in V$ with respect to edge weights $\bar{w}_{ij} = 1 - \bar{x}_i - \bar{x}_j$ for every $ij \in E$ have to be known. Thus, we use the separation algorithm for odd cycles to determine such weights of shortest odd and even walks (the latter having at least two edges), respectively, from vertex u to vertex v in G. We denote them by $\bar{f}_{uv}$ and $\bar{g}_{uv}$, respectively. Let $P^1_{u,v}$ be a shortest odd walk between vertices u and v for fixed $\bar{x}$. Then $\mathring{P}^1_{u,v}$ is defined as the set of its interior vertices, i.e., the vertices on this path except the start and end point. Similarly, for a shortest *even* walk $P^0_{u,v}$ we denote the set of interior vertices by $\mathring{P}^0_{u,v}$. We can determine $\bar{f}$ and $\bar{g}$ via

$$\bar{f}_{uv} = \sum_{ij \in P^1_{u,v}} \bar{w}_{ij} = 1 - \bar{x}_u - \bar{x}_v + |\mathring{P}^1_{u,v}| - 2 \sum_{i \in \mathring{P}^1_{u,v}} \bar{x}_i \quad \text{and}$$

$$\bar{g}_{uv} = \sum_{ij \in P^0_{u,v}} \bar{w}_{ij} = 1 - \bar{x}_u - \bar{x}_v + |\mathring{P}^0_{u,v}| - 2 \sum_{i \in \mathring{P}^0_{u,v}} \bar{x}_i.$$

We consider the categorical product of the underlying graph G for the maximum stable set problem and F, where F is the digraph with vertex set $\{0, 1, 2, 3, 4, 5\}$ and arc set $\{(0,1), (1,2), (2,1), (2,3), (3,4), (4,5), (5,4), (5,0), (0,3), (3,0)\}$, see Figure 3.5.

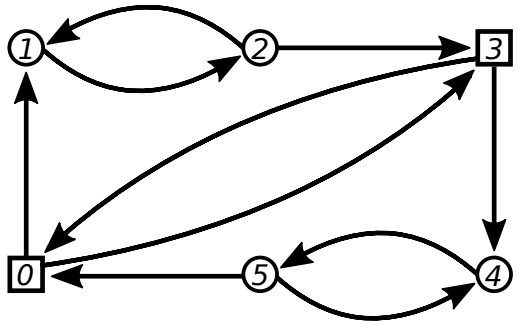


Figure 3.5: Digraph F for the construction of D_h.

Then, for every fixed $h \in V$ as a candidate for the hub of a 1-wheel, $D_h = (V_h, \Gamma_h) := G \times F$ consists of arcs $((u,i),(v,j))$ for every edge uv in G and every arc (i,j) in F.

Notice that a walk Z in D_h induces simultaenous walks Z_G in G and Z_F in F. The walk Z_G is just a usual walk and is going to represent the entire rim $C \dot{\cup} R$ of the 1-wheel while the walk Z_F makes sure that all the parity conditions on the rim are fulfilled. Spoke ends of a wheel are represented in F by the vertices 0 and 3. A walk of a single edge between two spoke ends uses in F the arc $(0,3)$ or $(3,0)$. On the other hand a truly subdivided walk between two spoke ends v_i and v_{i+1} yields in F a walk from 0 to 1 to 2 (then maybe a couple times back to 1 and 2) and finally to 3 or similarily in F from 3 via 4 and 5 to 0.

Next we define weights as by de Vries (2015), that ensure that the weight of a rim walk is represented correctly in this product.

For every vertex $h \in V$ and arc $((u,i),(v,j)) \in \Gamma_h$ we define the following arc weights $\bar{w}^{\mathcal{A}h}_{uivj}$ and $\bar{w}^{\mathcal{B}h}_{uivj}$:

$$\left.\begin{aligned}
\bar{w}^{\mathcal{A}h}_{u0v3} = \bar{w}^{\mathcal{A}h}_{u3v0} &= \bar{f}_{hu} + \bar{f}_{hv} - \bar{x}_u - \bar{x}_v \\
\bar{w}^{\mathcal{A}h}_{u0v1} = \bar{w}^{\mathcal{A}h}_{u3v4} &= \bar{f}_{hu} + 1 - \bar{x}_u - 2\bar{x}_v - \bar{x}_h \\
\bar{w}^{\mathcal{A}h}_{u2v3} = \bar{w}^{\mathcal{A}h}_{u5v0} &= \bar{f}_{hv} + 1 - 2\bar{x}_u - \bar{x}_v + \bar{x}_h \\
\bar{w}^{\mathcal{A}h}_{u1v2} = \bar{w}^{\mathcal{A}h}_{u2v1} = \bar{w}^{\mathcal{A}h}_{u4v5} = \bar{w}^{\mathcal{A}h}_{u5v4} &= 2(1 - \bar{x}_u - \bar{x}_v)
\end{aligned}\right\} \tag{3.1}$$

$$\left.\begin{aligned}
\bar{w}^{\mathcal{B}h}_{u0v3} = \bar{w}^{\mathcal{B}h}_{u3v0} &= \bar{g}_{hu} + \bar{g}_{hv} - \bar{x}_u - \bar{x}_v \\
\bar{w}^{\mathcal{B}h}_{u0v1} = \bar{w}^{\mathcal{B}h}_{u3v4} &= \bar{g}_{hu} + 1 - \bar{x}_u - 2\bar{x}_v - \bar{x}_h \\
\bar{w}^{\mathcal{B}h}_{u2v3} = \bar{w}^{\mathcal{B}h}_{u5v0} &= \bar{g}_{hv} + 1 - 2\bar{x}_u - \bar{x}_v + \bar{x}_h \\
\bar{w}^{\mathcal{B}h}_{u1v2} = \bar{w}^{\mathcal{B}h}_{u2v1} = \bar{w}^{\mathcal{B}h}_{u4v5} = \bar{w}^{\mathcal{B}h}_{u5v4} &= 2(1 - \bar{x}_u - \bar{x}_v)
\end{aligned}\right\} \tag{3.2}$$

The superscript $\mathcal{A}h$ refers to the 1-wheel inequalities of type $(I'_{\mathcal{A}})$ with h fixed as a hub and $\mathcal{B}h$ refers to $(I'_{\mathcal{B}})$. The weights of all arcs but those of the fourth type in (3.1) and (3.2) depend on h. These weights are taken from the arc weight construction of de Vries (2015), in order to state the following technical lemma that will be central in the proof of our main theorem.

Lemma 3.4. *(de Vries, 2015, Corollary 11) For a graph G and $\bar{x} \in P^{OC}(G)$, there is a violated inequality $(I'_{\mathcal{A}})$ with hub h and rim starting in v if and only if D_h contains a walk from $(v,0)$ to $(v,3)$ of weight less than $2 - 2\bar{x}_h$ with respect to $\bar{w}^{\mathcal{A}h}$. There is a violated inequality $(I'_{\mathcal{B}})$ with hub h and rim starting in v if and only if D_h contains a walk from $(v,0)$ to $(v,3)$ of weight less than $2\bar{x}_h$ with respect to $\bar{w}^{\mathcal{B}h}$.*

As mentioned, the representation property in Lemma 2.16 ensures that satisfying all $(I'_{\mathcal{A}})$-inequalities and all $(I'_{\mathcal{B}})$-inequalities is sufficient to satisfy all $(I_{\mathcal{A}})$-inequalities and all $(I_{\mathcal{B}})$-inequalities for 1-wheels without spokes of length 1.

3.3.2 The algorithm of Cheng and Cunningham

The separation algorithm of Cheng and Cunningham (1997) is competitive with the algorithm of de Vries (2015) for dense graphs. However, there are direct combinatorial methods that are fast in practice for dense graphs with $m = \Omega(n^2)$, since finding a maximum stable set in a given graph G is equivalent to finding a maximum clique in its complementary graph $\bar{G}$. A literature overview on efficient algorithms for the maximum clique problem, especially when applied to sparse graphs, is presented in Chapter 7. Nevertheless, we recapitulate the separation algorithm of Cheng and Cunningham for the $(I_{\mathcal{A}})$-inequalities in this section and additionally apply the idea of representing every $(I_{\mathcal{A}})$-inequality by an $(I'_{\mathcal{A}})$-inequality, see Lemma 2.16. This allows for constructing a more compact extended formulation in

Chapter 4. A separation algorithm for the $(I_{\mathcal{B}})$-inequalities can be constructed similarly. Our notation will slightly deviate from Cheng and Cunningham's.

Let $\bar{x} \in P^{OC}(G)$ and again $\bar{w}_{ij} = 1 - \bar{x}_i - \bar{x}_j$ for every $ij \in E$. For every pair $u, v \in V$, we compute the weight of shortest odd and even walks, respectively, from vertex u to v and denote them by $\bar{f}_{uv}$ and $\bar{g}_{uv}$. Let $h \in V$ be fixed. We now show how to ensure that every $(I_{\mathcal{A}})$-inequality for 1-wheels with hub h is fulfilled by $\bar{x}$. First, a complete auxiliary graph $H = (V_H, E_H)$ with loops is constructed where $V_H \coloneqq V^{\mathcal{O}} \cup V^{\mathcal{E}}$ is the union of two disjoint copies of V. In general, 1-wheels consist of odd and even spoke ends. The vertex set $V^{\mathcal{O}}$ represents all potential odd spoke ends, whereas $V^{\mathcal{E}}$ represents all potential even spoke ends. Since all face cycles of a simple 1-wheel are odd, the parity of the rim path between spoke ends v_i and v_{i+1} depends on their belonging to $V^{\mathcal{O}}$ and $V^{\mathcal{E}}$, respectively. For instance, if v_i is an odd spoke end and v_{i+1} is an even spoke end of some simple 1-wheel W in G, the rim path that belongs to the cycle $h - \ldots - v_i - \ldots - v_{i+1} - \ldots - h$ in W must be even. This gives an impression on why there are three different types of edge weights to define. Thus, weights

$$\bar{w}^h_{ij} = \begin{cases} \frac{1}{2}\bar{f}_{hi} + \frac{1}{2}\bar{f}_{hj} + \bar{f}_{ij} + \frac{1}{2}\bar{x}_i + \frac{1}{2}\bar{x}_j - 1 & \forall\, i,j \in V^{\mathcal{O}} \\ \frac{1}{2}\bar{g}_{hi} + \frac{1}{2}\bar{g}_{hj} + \bar{f}_{ij} - \frac{1}{2}\bar{x}_i - \frac{1}{2}\bar{x}_j & \forall\, i,j \in V^{\mathcal{E}} \\ \frac{1}{2}\bar{g}_{hi} + \frac{1}{2}\bar{f}_{hj} + \bar{g}_{ij} - \frac{1}{2}\bar{x}_i + \frac{1}{2}\bar{x}_j - \frac{1}{2} & \forall\, i \in V^{\mathcal{E}}, j \in V^{\mathcal{O}} \end{cases}$$

are defined for the edges $ij \in E_H$. They have a very useful property which is important to be able to apply shortest path algorithms to H.

Lemma 3.5. *(Cheng and Cunningham, 1997, Lemma 3.6) All edge weights in H are nonnegative, i.e. $\bar{w}^h_{ij} \geq 0$ for all $ij \in E_H$.*

As mentioned above, H is a complete graph with loops. Usually, loops are not important for shortest paths or walks if their weight is nonnegative. For our purposes, the parity of a walk is relevant and sometimes a loop can be used to change the parity cheap enough to be part of the walk. The next lemma is very important for Theorem 3.7.

Lemma 3.6. *(Cheng and Cunningham, 1997, Lemma 3.7) For every loop $ii \in E_H$, the inequality $\bar{w}^h_{ii} \geq 1 - \bar{x}_h$ holds.*

The following theorem states an equivalent condition for the auxiliary graph H to satisfying all $(I_{\mathcal{A}})$-inequalities where h is the hub of the respective 1-wheel. It enables to solve the separation problem for all $(I_{\mathcal{A}})$-inequalities in polynomial time.

Theorem 3.7. *(Cheng and Cunningham, 1997, Theorem 3.8) If C is a minimum-weight odd cycle in H, then no inequality of the form $(I_{\mathcal{A}})$ with hub h is violated by $\bar{x}$ if and only if $\bar{w}^h(E_C) \geq 1 - \bar{x}_h$.*

Applying this procedure for every $h \in V$ as a candidate for a hub of a 1-wheel solves the separation problem.

Lemma 2.16 allows for saving some running time. Every $(I_{\mathcal{A}})$-inequality can be represented by an $(I'_{\mathcal{A}})$-inequality. Thus, it is not necessary to treat any vertex as a potential even spoke end since we may assume $\mathcal{E} = \emptyset$. The vertex set $V_H = V^{\mathcal{O}}$ of the complete auxiliary graph H with loops now is just one copy of V. Accordingly, we only need the first type of edge weights $\bar{w}^h_{ij}$ from above in H, that is

$$\bar{w}^h_{ij} = \frac{1}{2}\bar{f}_{hi} + \frac{1}{2}\bar{f}_{hj} + \bar{f}_{ij} + \frac{1}{2}\bar{x}_i + \frac{1}{2}\bar{x}_j - 1 \qquad \forall\, i, j \in V_H.$$

3.4 More separation algorithms and heuristics

Beyond the previous results, which are important for the extended formulations in Chapter 4, there exist a lot of more separation algorithms for various classes of stable set inequalities.

A polynomial time separation algorithm for a relaxation of the stable set polytope including all clique, odd cycle, odd antihole, and 1-wheel inequalities for simple 1-wheels where $S = R = \emptyset$ is given by Lovász and Schrijver (1991). Cheng (1998) shows how to separate the so-called generalized bicycle wheel inequalities of the cut polytope, which are closely related to the 1-wheel inequalities. He demonstrates the strength of the generalized bicycle wheel inequalities beyond the odd cycle inequalities with numerical experiments. Although he applies separation algorithms to the cut polytope, these results are relevant for the stable set polytope, since there exists a transformation between valid inequalities for the cut polytope and the stable set polytope. Giandomenico and Letchford (2006) present a separation algorithm for all web inequalities. The antiweb inequalities and the 1-wheel inequalities are extended to the so-called antiweb-wheel inequalities by Cheng and de Vries (2002). Additionally, they give a separation algorithm for them.

Although we do not consider separation heuristics here, we want to mention some of them that can be very efficient. Nemhauser and Sigismondi (1992) present a strong algorithm using cutting planes from clique and lifted odd hole inequalities. A separation heuristic for the rank inequalities is presented by Rossi and Smriglio (2001).

Giandomenico et al. (2009, 2013) and Rebennack et al. (2011) show how to generate strong cutting planes and propose branch-and-cut algorithms for the maximum stable set problem. They substantiate their strength with extensive computational results.

Chapter 4

Extended formulations of stable set relaxations

In this chapter, we present extended formulations of the odd cycle polytope $P^{OC}(G)$ and for the 1-wheel polytope $P^{W}(G)$. We compare the number of variables and inequalities of our new extended formulation of $P^{OC}(G)$ to two other formulations and demonstrate its strength by our computational results. Moreover, we construct two different formulations for the 1-wheel polytope $P^{W}(G)$, which arise from the separation algorithm of de Vries (2015) and that are fast for sparse or medium dense graphs. Additionally, an extended formulation of the odd 1-wheel polytope $P^{W_{\mathcal{A}}}(G)$, that arises from the separation algorithm of Cheng and Cunningham (1997), is presented. The intersection of the extended formulation of $P^{W_{\mathcal{A}}}(G)$ with an extended formulation of $P^{W_{\mathcal{B}}}(G)$, which can be constructed analogously, is another extended formulation of $P^{W}(G)$. We prefer this formulation for dense graphs.

4.1 Extended formulations

In an instance of the linear integer optimization problem, a well approximating LP relaxation may have an exponential number of inequalities. The idea of an extended formulation is to add new variables to the original variables and construct an LP with a *polynomial* number of linear constraints, which can be used to find an optimal solution to the original LP. If an extended formulation for a given LP exists, the projection of one of its optimal solutions in the original variables yields an optimal solution to the original LP.

A detailed introduction to extended formulations is given by Conforti et al. (2010), which we highly recommend to gain deep insights into this topic.

Definition 4.1. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron. The polyhedron $Q \subseteq \mathbb{R}^{n+m}$ is an* extended formulation *of P if there exists a projection of Q into $\mathbb{R}^n$ yielding P.*

For illustration, we visualize the benefits extended formulations potentially have by an example.

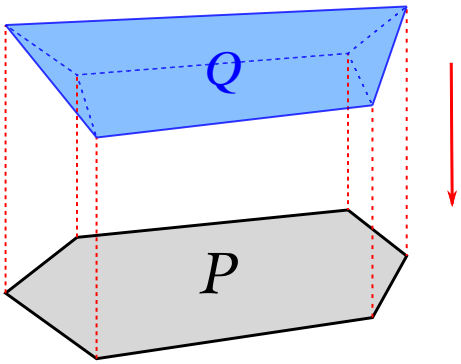


Figure 4.1: P is the projection of Q.

Example 4.2. *Figure 4.1 shows a polyhedron* $P \subseteq \mathbb{R}^2$ *with* six *facets. The polyhedron* $Q \subseteq \mathbb{R}^3$ *has only* five *facets and its projection into* $\mathbb{R}^2$*, depicted by the red arrow, yields* P*.*

If a given LP has polynomially many variables and inequalities, it can be solved in polynomial time via the ellipsoid method, see Grötschel et al. (1981). Although many classes of valid inequalities of the stable set polytope are known that have exponential size, there exist polynomial time separation algorithms for some of them. Carr and Lancia (2002) prove that for an exponential size LP, optimization with a polynomial size extended formulation is possible if and only if it is possible to state a separation algorithm as a polynomial size LP.

A method for generating linear extended formulations by using cutting plane methods is developed by Martin (1991), who beyond that gives the first polynomial size formulation of the spanning tree problem. For an extensive insight into the theoretical background of problems it can be applied to, we refer the reader to Lancia and Serafini (2014). Moreover, Kaibel (2011) presents results on extension techniques and lower bounds on the extension size.

The method of Martin is used for various combinatorial optimization problems. Goemans and Myung (1993), for example, apply it to Steiner tree formulations. Lancia and Serafini (2011) use this approach to develop a compact extended formulation for the max cut problem. Quadratic size extended formulations for independence polytopes of graphic and cographic matroids by application of this method are presented by Kaibel et al. (2016).

4.2 The odd cycle polytope

In this section, we present our results from the article *"A smaller extended formulation for the odd cycle inequalities of the stable set polytope"*, see de Vries and Perscheid (2020). We compare three different extended formulations that can be used to optimize over $P^{OC}(G)$ in polynomial time. The first one arises naturally from the separation procedure in Section 3.2 in combination with the ideas of Section 3.1.2. The second one is the well-known extended

formulation from Yannakakis (1991). The third formulation is more efficient as it combines the strengths of the definition of the other two formulations.

4.2.1 A direct approach

A natural approach to obtain an extended formulation of the odd cycle polytope $P^{OC}(G)$ is to use the idea at the end of Section 3.1.2 and adapt it to every pair $(i,0)$ and $(i,1)$ with $i \in V$ in the categorical graph product $G \times H$ from Section 3.2, where $H = (V_H, E_H)$ with $V_H = \{0,1\}$ and $E_H = \big\{\{0,1\}\big\}$. The linear program for solving shortest path problems requires a digraph, hence we replace every edge by two antiparallel arcs. We define $A_{G\times H} \coloneqq \big\{((i,r),(j,s)) : \{(i,r),(j,s)\} \in E_{G\times H}\big\}$. For every edge ij in G, we have $\{(i,0),(j,1)\}$ and $\{(i,1),(j,0)\}$ in $E_{G\times H}$ and hence $|E_{G\times H}| = 2m$. Thus, $|A_{G\times H}| = 4m$. Since shortest $(i,0)$-$(i,1)$-paths in $G \times H$ have the same weight as shortest odd cycles in G through i, they should have weight at least 1, see Corollary 3.3. Thus, with $c = 1$ in (F-SP) from Section 3.1.2:

$$Q_0^{OC}(G) \coloneqq \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^{4n^2} : (x,y) \text{ satisfies (2.1), (2.2), and (4.1)–(4.3)}\}$$

with

$$\begin{aligned}
&y_{irir} = 0 && \forall\ i \in V,\ r \in \{0,1\} && (4.1)\\
&y_{irjs} \le y_{irkt} + w_{ktjs} && \forall\ ((k,t),(j,s)) \in A_{G\times H},\ i \in V,\ r \in \{0,1\} && (4.2)\\
&y_{i0i1} \ge 1 && \forall\ i \in V && (4.3)
\end{aligned}$$

is an extended formulation of $P^{OC}(G)$.

We compare the number of variables and inequalities of all extended formulations that are presented in this section. If we fix variables as constants, such as the y_{irir} in equations (4.1), we do not count them as variables and the respective equations are not counted as inequalities.

Remark 4.3. *The extended formulation $Q_0^{OC}(G)$ requires $n + 4n^2 - 2n = 4n^2 - n$ variables and has $n + m + 8mn + n = 8mn + m + 2n$ inequalities.*

4.2.2 Yannakakis' formulation

The following polyhedron is a well-known extended formulation of $P^{OC}(G)$, due to Yannakakis (1991), which uses less variables and inequalities than $Q_0^{OC}(G)$. For easier counting of inequalities, we define $A \coloneqq \{(i,j) : ij \in E\}$ and use it for inequalities (4.4)–(4.6) in contrast to the original formulation of Yannakakis (if, for example, variable f_{ij} is involved for edge $ij \in E$, then ij simultaneously produces an inequality for variable f_{ji}). Then $|A| = 2m$, since for every edge ij we have $(i,j) \in A$ and $(j,i) \in A$.

$$Q_1^{OC}(G) \coloneqq \{(x,f,g) \in \mathbb{R}^n \times \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} : (x,f,g) \text{ satisfies (2.1) and (4.4)–(4.7)}\}$$

with

$$0 \leq f_{ij} \leq 1 - x_i - x_j \qquad \forall\ (i,j) \in A \tag{4.4}$$
$$f_{ij} \leq f_{ik} + g_{kj} \qquad \forall\ (i,k) \in A,\ j \in V \tag{4.5}$$
$$g_{ij} \leq f_{ik} + f_{kj} \qquad \forall\ (i,k) \in A,\ j \in V \tag{4.6}$$
$$f_{ii} \geq 1 \qquad \forall\ i \in V. \tag{4.7}$$

The edge inequalities (2.2) are implied by inequalities (4.4) in $Q_1^{OC}(G)$, because $0 \leq 1 - x_i - x_j$ is equivalent to $x_i + x_j \leq 1$ for all edges $ij \in E$. Inequalities (4.4), (4.5), and (4.6) imply that f_{ij} is bounded from above by the weight of a shortest *odd* walk between two vertices i and j, whereas g_{ij} is bounded from above by the weight of a shortest *even* walk using at least two edges between vertices i and j. Finally, inequalities (4.7) ensure that all odd cycle inequalities hold.

Remark 4.4. *Yannakakis' formulation $Q_1^{OC}(G)$ from above requires $2n^2 + n$ variables and has $n + 4m + 2mn + 2mn + n = 4mn + 4m + 2n$ inequalities.*

4.2.3 A smaller extended formulation

Analogously to $Q_1^{OC}(G)$, we add variables $f, g \in \mathbb{R}^{n^2}$ to the original variables $x \in \mathbb{R}^n$ for our new extended formulation $Q_2^{OC}(G)$ and define $A \coloneqq \{(i,j) : ij \in E\}$. The set of (in)equalities of $Q_2^{OC}(G)$ combines the benefits of $Q_0^{OC}(G)$ and $Q_1^{OC}(G)$ with the idea that the variables that represent the weight of shortest even closed walks can be fixed to 0.

$$Q_2^{OC}(G) \coloneqq \{(x,f,g) \in \mathbb{R}^n \times \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} : (x,f,g) \text{ satisfies (2.1) and (4.8)–(4.11)}\}$$

with

$$g_{ii} = 0 \qquad \forall\ i \in V \tag{4.8}$$
$$f_{ij} \leq g_{ik} + 1 - x_k - x_j \qquad \forall\ (k,j) \in A,\ i \in V \tag{4.9}$$
$$g_{ij} \leq f_{ik} + 1 - x_k - x_j \qquad \forall\ (k,j) \in A,\ i \in V \tag{4.10}$$
$$f_{ii} \geq 1 \qquad \forall\ i \in V. \tag{4.11}$$

Notice that the edge inequalities are automatically fulfilled in the formulation above. However, this is less obvious than in the formulation $Q_1^{OC}(G)$:

Lemma 4.5. *The edge inequalities* (2.2) *are implied by the set of inequalities of $Q_2^{OC}(G)$.*

Proof. Let $uv \in E$ be some edge for which we want to show that $x_u + x_v \leq 1$ holds. Consider inequalities (4.9) with $i = k = u$ and $j = v$. Then we have

$$f_{uv} \leq g_{uu} + 1 - x_u - x_v.$$

For inequalities (4.10), let $i = j = u$ and $k = v$. Thus,

$$g_{uu} \leq f_{uv} + 1 - x_v - x_u.$$

With $g_{uu} = 0$ by equations (4.8), adding both inequalities from above yields

$$f_{uv} \leq f_{uv} + 2(1 - x_u - x_v),$$

which simplifies to

$$x_u + x_v \leq 1. \qquad \square$$

Theorem 4.6. *$Q_2^{OC}(G)$ is an extended formulation of $P^{OC}(G)$.*

Proof. Let $\bar{x} \in P^{OC}(G)$. We show that there exist $\bar{f}$ and $\bar{g}$ so that $(\bar{x}, \bar{f}, \bar{g}) \in Q_2^{OC}(G)$ holds. All inequalities (2.1) occur in both polytopes and they are not violated by $\bar{x}$. Assign the nonnegative weight $\bar{w}_{ij} = 1 - \bar{x}_i - \bar{x}_j$ to every edge $ij \in E$ and hence to every arc $(i, j) \in A$. Furthermore, define $\bar{f}_{ij}$ for all $i, j \in V$ and $\bar{g}_{ij}$ for all $i, j \in V$ as the weights of shortest odd and even walks, respectively, between vertices i and j in G (assign a large value to the corresponding variable if no such walk exists). Therefore, the variables f and g are symmetric, i.e. $\bar{f}_{ij} = \bar{f}_{ji}$ and $\bar{g}_{ij} = \bar{g}_{ji}$ for all $i, j \in V$. Notice that, because there are no cycles with negative weight, a shortest even walk for every $i \in V$ to itself can simply use zero edges and therefore has weight 0. This ensures that equations (4.8) hold. It is easy to see that inequalities (4.9) and (4.10) are not violated either due to the construction of $\bar{f}$ and $\bar{g}$. These inequalities can be interpreted as follows: The weight of shortest odd or even walks from vertex i to j cannot exceed the weight of a shortest walk of the opposite parity from i to k plus the weight of the arc (k, j).

Inequalities (2.3) are satisfied by $\bar{x}$. This is, as mentioned above, equivalent to every odd cycle C having weight $\bar{w}(E_C)$ at least 1. Observe that $\bar{f}_{ii}$ is the weight of a shortest odd closed *walk* starting in $i \in V$. If this walk is not a cycle, there always exists an odd cycle with weight less or equal to the weight of a shortest odd closed walk. However, this cycle does not necessarily include vertex i, but its weight has to be as well greater or equal to 1. Therefore $\bar{f}_{ii} \geq 1$ holds for every $i \in V$.

For the converse, let $(\bar{x}, \bar{f}, \bar{g}) \in Q_2^{OC}(G)$. No edge inequality is violated by $\bar{x}$, see Lemma 4.5. Inequalities (4.8), (4.9), and (4.10) ensure that $\bar{f}_{ij}$ and $\bar{g}_{ij}$ are bounded from above by the weights of shortest odd and even walks, respectively, between vertices i and j. Thus, the value $\bar{f}_{ii}$ is lower or equal than the weight of a shortest odd cycle through vertex i. Since $\bar{f}_{ii} \geq 1$ for every $i \in V$ by inequalities (4.11), no odd cycle inequality is violated and hence $\bar{x} \in P^{OC}(G)$. $\square$

Remark 4.7. *Formulation $Q_2^{OC}(G)$ requires just $n + 2n^2 - n = 2n^2$ variables and $n + 2mn + 2mn + n = 4mn + 2n$ inequalities.*

4.2.4 Comparison of $Q_1^{OC}(G)$ and $Q_2^{OC}(G)$

Both formulations $Q_1^{OC}(G)$ and $Q_2^{OC}(G)$ use the same variables f and g and they are extended formulations of $P^{OC}(G)$. One wonders about their relation: Is one polyhedron a subset of the other one? Before we show why this is not the case, we point out one special property of the variables g_{ll} in $Q_1^{OC}(G)$.

Lemma 4.8. *Let $(\bar{x}, \bar{f}, \bar{g}) \in Q_1^{OC}(G)$ and $\hat{g}_{ij} := \bar{g}_{ij}$ for all $i, j \in V$ with $i \neq j$. Then for all $\hat{g}_{ll}$ in the interval $[0, 2\min\{\bar{f}_{lk} : (l,k) \in A\}]$ with $l \in V$ we have $(\bar{x}, \bar{f}, \hat{g}) \in Q_1^{OC}(G)$.*

Proof. For each $l \in V$ the variable g_{ll} occurs in inequalities (4.5), i.e. $f_{il} \leq f_{il} + g_{ll}$ if $j = k = l$. Therefore it is bounded from below by zero and there is no tighter lower bound given by the set of inequalities. On the other hand the variable g_{ll} is bounded from above by inequalities (4.6) when $i = j = l$. In this case we obtain $g_{ll} \leq f_{lk} + f_{kl}$ for every $(l,k) \in A$. We have $2\min\{f_{lk}, f_{kl}\} \leq f_{lk} + f_{kl}$ for all $(l,k) \in A$ and there is no upper bound that is smaller than $f_{lk} + f_{kl}$. Therefore $(\bar{x}, \bar{f}, \hat{g}) \in Q_1^{OC}(G)$. □

Theorem 4.9. *In general, $Q_1^{OC}(G) \not\subseteq Q_2^{OC}(G)$.*

Proof. Let w.l.o.g. $\bar{x} = \mathbf{0}$ and $\bar{f} = \mathbf{1}$. Then $\bar{g}_{ll} = 2$ is feasible for all $l \in V$, since $\bar{g}_{ll} = 2 \in [0, 2\min\{\bar{f}_{lk} : (l,k) \in A\}] = [0, 2]$, cf. Lemma 4.8. All the other variables $\bar{g}_{ij}$ with $i \neq j$ can attain every value from the interval $[0, 2]$. It follows that $(\bar{x}, \bar{f}, \bar{g}) \in Q_1^{OC}(G)$. Moreover, $\bar{g}_{ll} = 2$ for all $l \in V$ violates inequalities (4.8) and thus $(\bar{x}, \bar{f}, \bar{g}) \notin Q_2^{OC}(G)$. □

Theorem 4.10. *In general, $Q_2^{OC}(G) \not\subseteq Q_1^{OC}(G)$.*

Proof. Consider $(\bar{x}, \bar{f}, \bar{g})$ with $\bar{x} = \mathbf{0}$, $\bar{g} = \mathbf{0}$, $\bar{f}_{ii} = 1$ for all $i \in V$, and $\bar{f}_{ij} = -1$ for all $i, j \in V$ with $i \neq j$. Then $(\bar{x}, \bar{f}, \bar{g}) \in Q_2^{OC}(G)$, but $(\bar{x}, \bar{f}, \bar{g}) \notin Q_1^{OC}(G)$, since inequalities (4.4) restrict variables f_{ij} to be nonnegative if $(i,j) \in A$. □

All the extended formulations presented in Sections 4.2.1, 4.2.2, and 4.2.3 are of polynomial size, although their efficiency differs significantly.

Corollary 4.11. *The MSSP for t-perfect graphs can be solved with each formulation $Q_0^{OC}(G)$, $Q_1^{OC}(G)$, and $Q_2^{OC}(G)$ in polynomial time.*

4.2.5 Numerical results

To complement the theoretical results for the extended formulations $Q_0^{OC}(G)$, $Q_1^{OC}(G)$, and $Q_2^{OC}(G)$ of the odd cycle polytope $P^{OC}(G)$, some numerical experiments are performed. We use CPLEX v. 12.8.0.0 as an LP solver for an extensive set of test instances. These instances are generated randomly with the graph generator *fast_gnp_random_graph* from the Python package NetworkX, see Hagberg et al. (2008). The input data are the number of vertices n and a probability p for each pair i and j of vertices in G to share an edge. We tabulate the graphs with respect to n and the expected density d (that equals p). For each pair of parameters n and d we consider, we compute the average CPLEX running time for optimizing over any extended formulation of $P^{OC}(G)$ on 100 different test instances.

By Remark 2.6, $\bar{x} = \left(\frac{1}{3}, \ldots, \frac{1}{3}\right)^\mathsf{T} \in P^{OC}(G)$ for any graph G. Since $\bar{x} \notin \mathrm{STAB}(G)$ if there exists some clique K in G with $|K| \geq 4$, as it violates the clique inequality for K, it is more likely that $\bar{x} \in \mathrm{STAB}(G)$ in sparse than in dense graphs. Therefore, the odd cycle polytope might be a better approximation of $\mathrm{STAB}(G)$ if G is sparse. For illustration, we report the optimality gap

$$\mathrm{gap}(z) \coloneqq \left| \frac{z^* - z}{z} \right| \times 100,$$

where z^* is the objective value of an optimal solution of the maximum stable set problem and z is the objective value of an optimal solution of some relaxation. For every set of test instances, the average gap of $z^E(G)$ and $z^{OC}(G)$, the objective values arising from optimizing over $P^E(G)$ and $P^{OC}(G)$, respectively, is additionally given in Tables 4.1, 4.2, and 4.3.

The polyhedron $Q_0^{OC}(G)$ has about twice as many variables as $Q_1^{OC}(G)$ and $Q_2^{OC}(G)$, respectively. Furthermore, its coefficient of the dominating term mn for the number of inequalities is twice that of the two other polyhedra. In a first step, we confirm the hypothesis that optimizing over $Q_0^{OC}(G)$ consumes much more running time than optimizing over each of the other polyhedra we consider.

Table 4.1: Average running time of $Q_0^{OC}(G)$, $Q_1^{OC}(G)$, and $Q_2^{OC}(G)$ with the barrier method; each number is computed as the average over 100 instances.

				Barrier method		
		Average gap (%)		Average CPU (s)		
n	d (%)	$z^E(G)$	$z^{OC}(G)$	$Q_0^{OC}(G)$	$Q_1^{OC}(G)$	$Q_2^{OC}(G)$
50	5	0.83	0.03	80.14	1.04	1.12
	10	15.09	0.61	79.89	2.08	2.11
	15	29.25	4.28	82.20	2.99	2.58
	20	40.24	12.58	84.99	3.82	3.48

The dual simplex method is the default method in CPLEX for solving linear programs. Unfortunately, even for $n = 50$ it was not possible to find optimal solutions with $Q_0^{OC}(G)$ in reasonable time. The barrier method (we permit up to 32 threads for parallelization for our experiments) turns out to be by far the best method available for our purposes. We can see in Table 4.1 that the average CPU time is many times higher for $Q_0^{OC}(G)$ than for $Q_1^{OC}(G)$ and $Q_2^{OC}(G)$, respectively. Therefore, we restrict our computational study to the running times of solving $Q_1^{OC}(G)$ and $Q_2^{OC}(G)$ thereafter.

Let $\mathrm{rt}(P^X(G))$ be the running time in seconds consumed by optimizing over the stable set relaxation $P^X(G)$. We define

$$\rho(G) \coloneqq \frac{\mathrm{rt}(Q_1^{OC}(G)) - \mathrm{rt}(Q_2^{OC}(G))}{\mathrm{rt}(Q_1^{OC}(G))} \times 100,$$

that displays how much time is proportionately saved by the relaxation $Q_2^{OC}(G)$ in comparison to the relaxation $Q_1^{OC}(G)$.

Table 4.2: Average running time of $Q_1^{QC}(G)$ and $Q_2^{QC}(G)$ with the dual simplex method; each number is computed as the average over 100 instances.

				Dual simplex method		
		Average gap (%)		Average CPU (s)		
n	d (%)	$z^E(G)$	$z^{QC}(G)$	$Q_1^{QC}(G)$	$Q_2^{QC}(G)$	$\rho(G)$
50	5	0.83	0.03	1.11	1.06	5 %
	10	15.09	0.61	18.52	3.15	83 %
	15	29.25	4.28	162.23	4.64	97 %
	20	40.24	12.58	563.51	5.88	99 %

We solved all the problems for $n = 50$ with the dual simplex method. Table 4.2 shows the average running time over 100 instances for every density $d \in \{5, 10, 15, 20\}$. We obtain that the dual simplex method gets really slow on larger instances when optimizing over $Q_1^{QC}(G)$. The worst running time among all single test instances consumed by optimizing over $Q_1^{QC}(G)$ is 1,152 seconds, whereas optimizing over $Q_2^{QC}(G)$ never took more than 11 seconds.

Table 4.3: Average running time of $Q_1^{QC}(G)$ and $Q_2^{QC}(G)$ with the barrier method; each number is computed as the average over 100 instances.

				Barrier method		
		Average gap (%)		Average CPU (s)		
n	d (%)	$z^E(G)$	$z^{QC}(G)$	$Q_1^{QC}(G)$	$Q_2^{QC}(G)$	$\rho(G)$
50	5	0.83	0.03	1.04	1.12	-8 %
	10	15.09	0.61	2.08	2.11	-1 %
	15	29.25	4.28	2.99	2.58	14 %
	20	40.24	12.58	3.82	3.48	9 %
75	5	5.65	0.18	4.13	3.82	8 %
	10	28.86	5.34	8.42	6.46	23 %
	15	43.23	16.52	12.33	9.43	24 %
	20	52.72	29.08	15.48	11.76	24 %
100	5	14.35	0.97	12.08	9.67	20 %
	10	38.82	13.41	22.95	16.05	30 %
	15	51.96	28.04	33.29	22.33	33 %
125	5	21.93	3.51	25.28	22.01	13 %
	10	45.68	20.21	50.96	37.88	26 %
	15	57.86	36.78	70.70	53.19	25 %
150	5	28.52	7.74	53.64	35.57	34 %
	10	50.95	26.75	97.90	59.84	39 %
	15	62.27	43.40	146.67	88.88	39 %
175	5	33.34	11.36	86.99	62.09	29 %
	10[1]	55.16	32.81	163.67	100.74	38 %
200	5	38.02	15.22	147.53	86.08	42 %
	10	58.84	38.28	275.46	139.82	49 %

As the dual simplex method turns out to be much slower than the barrier method for optimizing over $Q_1^{QC}(G)$ and $Q_2^{QC}(G)$, respectively, we tried to solve all problems for $n \in$

[1]Three test instances were excluded, because the time limit of 3,600 seconds was reached while solving $Q_1^{QC}(G)$. Nevertheless, $Q_2^{QC}(G)$ was solved in less than 90 seconds for each of the excluded instances.

$\{50, 75, 100, 125, 150, 175, 200\}$ with the barrier method. Up to 32 threads are used for parallelization and the time limit is set to 3,600 seconds. The results are given in Table 4.3. The time limit was only exceeded by three test instances with $n = 175$ when optimizing over $Q_1^{OC}(G)$. Among all remaining test instances that were solved with the barrier method, the worst running time was 307 seconds when using $Q_1^{OC}(G)$ and 212 seconds when using $Q_2^{OC}(G)$.

To conclude, the barrier method performs better than the dual simplex method on both formulations $Q_1^{OC}(G)$ and $Q_2^{OC}(G)$. Furthermore, optimizing over $Q_2^{OC}(G)$ is much faster than optimizing over $Q_1^{OC}(G)$.

4.3 The 1-wheel polytope

Two different separation algorithms for the 1-wheel inequalities of the stable set polytope are described in Chapter 3. Except for dense graphs, the separation algorithm of de Vries (2015) is preferable as it has the better asymptotic runtime bound. This section provides two alternative compact extended formulations of the 1-wheel polytope $P^W(G)$ that arise from the algorithm presented in Section 3.3.1. Moreover, we show how to construct a compact extended formulation based on the separation algorithm of Cheng and Cunningham (1997), which is recapitulated in Section 3.3.2. This separation algorithm refers to the *odd* 1-wheel inequalities and therefore the formulation we present is an extended formulation of the odd 1-wheel polytope $P^{W_\mathcal{A}}(G) \supseteq P^W(G)$.

4.3.1 An extended formulation for arbitrary graphs

In what follows, the core results of our manuscript *"An extended formulation for the 1-wheel inequalities of the stable set polytope"*, see de Vries et al. (2019), are presented. Besides the formulation $Q_1^W(G)$, which is introduced and analyzed in our article, we additionally introduce an alternative formulation $Q_2^W(G)$. We will see that the construction of $Q_1^W(G)$ is related to the formulation $Q_1^{OC}(G)$ for the odd cycle polytope, whereas $Q_2^W(G)$ follows the notion of our extended formulation $Q_2^{OC}(G)$ from Section 4.2.3. Both formulations $Q_1^W(G)$ and $Q_2^W(G)$ are of polynomial size and imply the odd cycle inequalities as well as the odd *and* even 1-wheel inequalities.

Remember that the 1-wheel polytope is defined as $P^W(G) = P^{W_\mathcal{A}}(G) \cap P^{W_\mathcal{B}}(G) \subseteq P^{OC}(G)$. Regarding the separation algorithm of de Vries, Lemma 3.4 requires $\bar{x} \in P^{OC}(G)$. Thus, we start with inequalities (2.1) and (4.4)–(4.7), which constitute the polynomial size extended formulation $Q_1^{OC}(G)$. Then we extend it further to derive $Q_1^W(G)$, an extended formulation of the 1-wheel polytope $P^W(G)$. For this purpose we introduce arc variables $w_{uivj}^{\mathcal{A}h}$ and $w_{uivj}^{\mathcal{B}h}$ for every vertex $h \in V$ and all $O(m)$ arcs $((u,i),(v,j)) \in \Gamma_h$ in the categorical graph product D_h. Just as for the variables w_{ij} in the extended formulation for odd cycles, they are defined as substitutes for larger expressions, which makes the representation of $Q_1^W(G)$ much shorter and clearer. We define them similarly to (3.1) and (3.2) dependent on x, f, and g:

$$\left.\begin{aligned} w^{\mathcal{A}h}_{u0v3} = w^{\mathcal{A}h}_{u3v0} &= \ f_{hu} + f_{hv} - x_u - x_v \\ w^{\mathcal{A}h}_{u0v1} = w^{\mathcal{A}h}_{u3v4} &= \ f_{hu} + 1 - x_u - 2x_v - x_h \\ w^{\mathcal{A}h}_{u2v3} = w^{\mathcal{A}h}_{u5v0} &= \ f_{hv} + 1 - 2x_u - x_v + x_h \\ w^{\mathcal{A}h}_{u1v2} = w^{\mathcal{A}h}_{u2v1} = w^{\mathcal{A}h}_{u4v5} &= w^{\mathcal{A}h}_{u5v4} = \ 2(1 - x_u - x_v) \end{aligned}\right\} \tag{4.12}$$

$$\left.\begin{aligned} w^{\mathcal{B}h}_{u0v3} = w^{\mathcal{B}h}_{u3v0} &= \ g_{hu} + g_{hv} - x_u - x_v \\ w^{\mathcal{B}h}_{u0v1} = w^{\mathcal{B}h}_{u3v4} &= \ g_{hu} + 1 - x_u - 2x_v - x_h \\ w^{\mathcal{B}h}_{u2v3} = w^{\mathcal{B}h}_{u5v0} &= \ g_{hv} + 1 - 2x_u - x_v + x_h \\ w^{\mathcal{B}h}_{u1v2} = w^{\mathcal{B}h}_{u2v1} = w^{\mathcal{B}h}_{u4v5} &= w^{\mathcal{B}h}_{u5v4} = \ 2(1 - x_u - x_v) \end{aligned}\right\} \tag{4.13}$$

The variables f and g in $Q_1^{OC}(G)$ are constrained by inequalities in x. As in Section 4.2.2, they are bounded from above by the weights of shortest odd and even walks, respectively, in G with respect to $w_{ij} = 1 - x_i - x_j$. Although the variables $w^{\mathcal{A}h}_{uivj}$ and $w^{\mathcal{B}h}_{uivj}$ can obviously take negative values for $(i,j) \notin \{(1,2),(2,1),(4,5),(5,4)\}$, de Vries (2015) shows that D_h contains no *cycle* with negative (total) weight.

Theorem 4.12. *Consider the 1-wheel polytope*

$$P^W(G) = \big\{x \in \mathbb{R}^n : x \text{ satisfies (2.1), (2.2), (2.3), } (I'_{\mathcal{A}}) \text{, and } (I'_{\mathcal{B}})\big\}.$$

Then an extended formulation $Q_1^W(G)$ of $P^W(G)$ is given by

$$\begin{aligned} Q_1^W(G) = \big\{&(x, f, g, p^{\mathcal{A}}, p^{\mathcal{B}}) \in \mathbb{R}^n \times \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} \times \mathbb{R}^{36n^3} \times \mathbb{R}^{36n^3} : \\ &(x, f, g, p^{\mathcal{A}}, p^{\mathcal{B}}) \text{ satisfies (2.1), (4.4)–(4.7), and (4.14)–(4.19)}\big\} \end{aligned}$$

with $w^{\mathcal{A}h}_{uivj}$ and $w^{\mathcal{B}h}_{uivj}$ defined as in (4.12) *and* (4.13) *and*

$$p^{\mathcal{A}h}_{uivj} \le w^{\mathcal{A}h}_{uivj} \qquad \forall\ ((u,i),(v,j)) \in \Gamma_h,\ h \in V \tag{4.14}$$

$$p^{\mathcal{B}h}_{uivj} \le w^{\mathcal{B}h}_{uivj} \qquad \forall\ ((u,i),(v,j)) \in \Gamma_h,\ h \in V \tag{4.15}$$

$$p^{\mathcal{A}h}_{uivj} \le p^{\mathcal{A}h}_{uiwk} + p^{\mathcal{A}h}_{wkvj} \qquad \forall\ ((u,i),(w,k)) \in \Gamma_h,\ (v,j) \in V_h,\ h \in V \tag{4.16}$$

$$p^{\mathcal{B}h}_{uivj} \le p^{\mathcal{B}h}_{uiwk} + p^{\mathcal{B}h}_{wkvj} \qquad \forall\ ((u,i),(w,k)) \in \Gamma_h,\ (v,j) \in V_h,\ h \in V \tag{4.17}$$

$$p^{\mathcal{A}h}_{v0v3} \ge 2 - 2x_h \qquad \forall\ v \in V,\ h \in V \tag{4.18}$$

$$p^{\mathcal{B}h}_{v0v3} \ge 2x_h \qquad \forall\ v \in V,\ h \in V. \tag{4.19}$$

Remark 4.13. *Our formulation $Q_1^W(G)$ requires $72n^3 + 2n^2 + n$ variables. Notice that $w^{\mathcal{A}h}$ and $w^{\mathcal{B}h}$ can be replaced by their definition involving only the variables x, f, and g. The number of inequalities is $n + 4m + 2mn + 2mn + n + 20mn + 20mn + 120mn^2 + 120mn^2 + n^2 + n^2 = 240mn^2 + 44mn + 2n^2 + 4m + 2n$.*

We now prove that the extended formulation $Q_1^W(G)$ of the 1-wheel polytope $P^W(G)$, as given above, is correct.

Proof of Theorem 4.12. We first show that $P^W(G) \subseteq \text{proj}_x(Q_1^W(G))$. Let $\bar{x} \in P^W(G)$. We construct $\bar{f}$, $\bar{g}$, $\bar{p}^{\mathcal{A}}$, and $\bar{p}^{\mathcal{B}}$ so that $(\bar{x}, \bar{f}, \bar{g}, \bar{p}^{\mathcal{A}}, \bar{p}^{\mathcal{B}}) \in Q_1^W(G)$: Define $\bar{f}_{ij}$ for all $i, j \in V$ and $\bar{g}_{ij}$ for all $i, j \in V$ with $i \neq j$ as the weights of shortest odd and even walks, respectively, between vertices i and j in G (if no such path exists, then assign a large value to the corresponding variable). In particular, this implies $\bar{f}_{ij} = \bar{f}_{ji}$ and $\bar{g}_{ij} = \bar{g}_{ji}$. With these definitions, inequalities (2.1), and (4.4)–(4.6) are fulfilled. Let $\bar{g}_{ll} = 2\min\{\bar{w}_{lk} : lk \in E\}$ for every vertex $l \in V$. Then $\bar{g}_{ll}$ is exactly the weight of a shortest even closed walk in G using at least two edges. Here, we use that $2\min\{\bar{w}_{lk} : lk \in E\} = 2\min\{\bar{f}_{lk} : lk \in E\}$ since $\bar{f}_{lk} = \bar{f}_{kl}$.

These weights occur as terms in some arc weights $\bar{w}^{\mathcal{B}h}_{uivj}$ with $h = u$ or $h = v$. Assigning the weights to the respective variables is feasible for $Q_1^{OC}(G)$ and for $Q_1^W(G)$ by Lemma 4.8. For every $h \in V$ and for every pair (u, i) and (v, j) of vertices in D_h define $\bar{p}^{\mathcal{A}h}_{uivj}$ as the weight of a shortest walk in D_h from (u, i) to (v, j) with respect to arc weights $\bar{w}^{\mathcal{A}h}$. Since there is no cycle of negative weight in D_h, the shortest walks exist although arc weights in D_h can be negative. Analogously, define $\bar{p}^{\mathcal{B}h}_{uivj}$ as the weight of a shortest walk in D_h from (u, i) to (v, j) with respect to arc weights $\bar{w}^{\mathcal{B}h}$. Then $\bar{p}^{\mathcal{A}h}_{uivj}$ and $\bar{p}^{\mathcal{B}h}_{uivj}$ fulfill inequalities (4.14)–(4.17).

Since $\bar{x} \in P^W(G)$ satisfies inequalities (2.3), which is equivalent to $\bar{w}(E_C) \geq 1$ for all odd cycles C, $\bar{f}_{ii} \geq 1$ for every $i \in V$. Therefore, inequalities (4.7) are not violated. Moreover, $\bar{x} \in P^W(G)$ does not violate any inequality $(I'_{\mathcal{A}})$ or $(I'_{\mathcal{B}})$. It follows directly that $\bar{p}^{\mathcal{A}h}_{v0v3} \geq 2 - 2\bar{x}_h$ and $\bar{p}^{\mathcal{B}h}_{v0v3} \geq 2\bar{x}_h$ for every $h, v \in V$ due to Lemma 3.4. Thus, the inequalities (4.18) and (4.19) are not violated. Hence, $(\bar{x}, \bar{f}, \bar{g}, \bar{p}^{\mathcal{A}}, \bar{p}^{\mathcal{B}}) \in Q_1^W(G)$ and therefore $P^W(G) \subseteq \text{proj}_x(Q_1^W(G))$.

Now it remains to show that $\text{proj}_x(Q_1^W(G)) \subseteq P^W(G)$. Let $(\bar{x}, \bar{f}, \bar{g}, \bar{p}^{\mathcal{A}}, \bar{p}^{\mathcal{B}}) \in Q_1^W(G)$. By inequalities (4.4)–(4.6) the variables f_{ij} and g_{ij} are bounded from above by the weight of a shortest odd and even walk, respectively, between vertices i and j in G. Moreover, inequality (4.7) is satisfied for every $i \in V$. Therefore, every odd cycle has weight at least 1. This is equivalent to the condition that no odd cycle inequality in x is violated by $\bar{x}$.

The variables $p^{\mathcal{A}h}_{uivj}$ are bounded from above by the weight of a shortest walk with respect to arc variables $w^{\mathcal{A}h}$ between vertices (u, i) and (v, j) in D_h by inequalities (4.16) together with inequalities (4.14). Analogously, the variables $p^{\mathcal{B}h}_{uivj}$ are bounded from above by the weight of a shortest walk between vertices (u, i) and (v, j) in D_h with respect to arc variables $w^{\mathcal{B}h}$ by inequalities (4.15) and (4.17). Since all inequalities (4.18) are satisfied by $(\bar{x}, \bar{f}, \bar{g}, \bar{p}^{\mathcal{A}}, \bar{p}^{\mathcal{B}})$, no $(I'_{\mathcal{A}})$-inequality in x is violated by $\bar{x}$ due to Lemma 3.4. Similarly, all inequalities (4.19) are satisfied by $(\bar{x}, \bar{f}, \bar{g}, \bar{p}^{\mathcal{A}}, \bar{p}^{\mathcal{B}})$. This implies that no $(I'_{\mathcal{B}})$-inequality in x is violated by $\bar{x}$. It follows that $\bar{x} \in P^W(G)$ and therefore $\text{proj}_x(Q_1^W(G)) \subseteq P^W(G)$. □

An alternative extended formulation of the 1-wheel polytope can be given by using a direct application of (F-SP) from Section 3.1.2 and adding the inequalities that arise for variables $p^{\mathcal{A}h}$ and $p^{\mathcal{B}h}$ to $Q_2^{OC}(G)$. The structure of the extension then is similar to the structure of $Q_2^{OC}(G)$. An advantage of this alternative formulation $Q_2^W(G)$ is that it has less variables and inequalities than $Q_1^W(G)$.

Theorem 4.14. *An alternative extended formulation of the 1-wheel polytope $P^W(G)$ is given by*

$$Q_2^W(G) = \big\{(x, f, g, p^{\mathcal{A}}, p^{\mathcal{B}}) \in \mathbb{R}^n \times \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} \times \mathbb{R}^{36n^3} \times \mathbb{R}^{36n^3} : \\ (x, f, g, p^{\mathcal{A}}, p^{\mathcal{B}}) \text{ satisfies (2.1), (4.8)–(4.11), and (4.20)–(4.25)}\big\}$$

with $w^{\mathcal{A}h}_{uivj}$ and $w^{\mathcal{B}h}_{uivj}$ defined as in (4.12) *and* (4.13) *and*

$$p^{\mathcal{A}h}_{uiui} = 0 \qquad \forall\ (u,i) \in V_h,\ h \in V \tag{4.20}$$
$$p^{\mathcal{B}h}_{uiui} = 0 \qquad \forall\ (u,i) \in V_h,\ h \in V \tag{4.21}$$
$$p^{\mathcal{A}h}_{uivj} \le p^{\mathcal{A}h}_{uiwk} + w^{\mathcal{A}h}_{wkvj} \qquad \forall\ ((w,k),(v,j)) \in \Gamma_h,\ (u,i) \in V_h,\ h \in V \tag{4.22}$$
$$p^{\mathcal{B}h}_{uivj} \le p^{\mathcal{B}h}_{uiwk} + w^{\mathcal{B}h}_{wkvj} \qquad \forall\ ((w,k),(v,j)) \in \Gamma_h,\ (u,i) \in V_h,\ h \in V \tag{4.23}$$
$$p^{\mathcal{A}h}_{v0v3} \ge 2 - 2x_h \qquad \forall\ v \in V,\ h \in V \tag{4.24}$$
$$p^{\mathcal{B}h}_{v0v3} \ge 2x_h \qquad \forall\ v \in V,\ h \in V. \tag{4.25}$$

We omit the proof of correctness for Theorem 4.14 here, as it would be straightforward to the proof of Theorem 4.12 in combination with the proof of Theorem 4.6.

Remark 4.15. *The extended formulation $Q_2^W(G)$ requires $n + 2n^2 + 72n^3 - n - 12n^2 = 72n^3 - 10n^2$ variables. Notice that $w^{\mathcal{A}h}$ and $w^{\mathcal{B}h}$ can be replaced by their definition that include the variables x, f, and g. The number of inequalities is $n + 2mn + 2mn + n + 120mn^2 + 120mn^2 + n^2 + n^2 = 240mn^2 + 4mn + 2n^2 + 2n$.*

4.3.2 An extended formulation for dense graphs

If the underlying graph is very dense and we want to optimize over the 1-wheel polytope $P^W(G)$ in polynomial time with a linear program, we can derive an extended formulation that is based on the separation algorithm from Cheng and Cunningham as presented in Section 3.3.2. We give such an extended formulation $Q^{W_{\mathcal{A}}}(G)$ of the *odd* 1-wheel polytope $P^{W_{\mathcal{A}}}(G)$. An extended formulation $Q^{W_{\mathcal{B}}}(G)$ of the *even* 1-wheel polytope $P^{W_{\mathcal{B}}}(G)$ can be constructed similarly and combining $Q^{W_{\mathcal{A}}}(G)$ and $Q^{W_{\mathcal{B}}}(G)$ yields an extended formulation of the 1-wheel polytope $P^W(G)$.

We define $A_H \coloneqq \{(i,j) : ij \in E_H\}$ for easier counting of inequalities in Theorem 4.16, where E_H is the edge set of the auxiliary graph $H = (V_H, E_H)$ from the separation algorithm in Section 3.3.2. Then $|A_H| = n^2$, since H is a complete graph with loops and we have $(i,j) \in A_H$ for every $i, j \in V_H$ including the case $i = j$.

Theorem 4.16. *An extended formulation $Q^{W_{\mathcal{A}}}(G)$ of the odd 1-wheel polytope $P^{W_{\mathcal{A}}}(G)$ is given by*

$$Q^{W_{\mathcal{A}}}(G) = \big\{(x, f, g, p, q) \in \mathbb{R}^n \times \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} \times \mathbb{R}^{n^3} \times \mathbb{R}^{n^3} : \\ (x, f, g, p, q) \text{ satisfies (2.1), (4.8)–(4.11), and (4.26)–(4.29)}\big\}$$

with

$$q^h_{ii} = 0 \qquad \forall\ i,h \in V \tag{4.26}$$

$$p^h_{ij} \leq q^h_{ik} + w^h_{kj} \qquad \forall\ (k,j) \in A_H,\ i,h \in V \tag{4.27}$$

$$q^h_{ij} \leq p^h_{ik} + w^h_{kj} \qquad \forall\ (k,j) \in A_H,\ i,h \in V \tag{4.28}$$

$$p^h_{ii} \geq 1 - x_h \qquad \forall\ i,h \in V, \tag{4.29}$$

where w^h_{kj} *is a shorthand notation for* $\frac{1}{2}f_{hk} + \frac{1}{2}f_{hj} + f_{kj} + \frac{1}{2}x_k + \frac{1}{2}x_j - 1$.

Proof. We first prove that $P^{W_\mathcal{A}}(G) \subseteq \text{proj}_x(Q^{W_\mathcal{A}}(G))$. For $\bar{x} \in P^{W_\mathcal{A}}(G)$, we can find $\bar{f}$, $\bar{g}$, $\bar{p}$, and $\bar{q}$ so that $(\bar{x}, \bar{f}, \bar{g}, \bar{p}, \bar{q}) \in Q^{W_\mathcal{A}}(G)$. Define $\bar{f}_{ij}$ and $\bar{g}_{ij}$ for all $i,j \in V$ as the weights of shortest odd and even walks, respectively, between vertices i and j in G. Assign a large value to the corresponding variable if no such walk exists. Observe that shortest even walks use zero edges. Due to symmetry we have $\bar{f}_{ij} = \bar{f}_{ji}$ and $\bar{g}_{ij} = \bar{g}_{ji}$. Inequalities (2.1) and (4.8)–(4.10) are fulfilled. Moreover, inequalities (4.11) are not violated since $\bar{x} \in P^{W_\mathcal{A}}(G)$ satisfies inequalities (2.3), which is equivalent to $\bar{w}(E_C) \geq 1$ for every odd cycle C in G and hence $\bar{f}_{ii} \geq 1$ for every $i \in V$ holds.

We define $\bar{p}^h_{ij}$ and $\bar{q}^h_{ij}$, respectively, as the weight of a shortest odd and even walk with respect to edge weights $\bar{w}^h$ in H between vertices i and j. These walks exist by the absence of negative edge weights, cf. Lemma 3.5, and the fact that H is a complete graph with loops. This choice of $\bar{p}^h_{ij}$ and $\bar{q}^h_{ij}$ is obviously feasible for inequalities (4.26)–(4.28). Theorem 3.7 implies that all $(I_\mathcal{A})$-inequalities are fulfilled if and only if $\bar{w}^h(E_C) \geq 1 - \bar{x}_h$ for every odd cycle C in H and every hub $h \in V$. This is equivalent to $\bar{p}^h_{ii} \geq 1 - \bar{x}_h$ for every $i \in V$ and every hub $h \in V$, i.e. inequalities (4.29) are fulfilled.

For the converse, we show that $\text{proj}_x(Q^{W_\mathcal{A}}(G)) \subseteq P^{W_\mathcal{A}}(G)$. Let $(\bar{x}, \bar{f}, \bar{g}, \bar{p}, \bar{q}) \in Q^{W_\mathcal{A}}(G)$. The variables f_{ij} and g_{ij} are bounded from above by the weight of a shortest odd and even walk, respectively, between vertices i and j in G. Inequalities (4.8)–(4.10) imply the edge inequalities, cf. Lemma 4.5, and inequalities (4.11) ensure that every odd cycle in G has weight at least 1. Thus, no odd cycle inequality (2.3) in x is violated by $\bar{x}$.

Furthermore, inequalities (4.26)–(4.28) ensure that p^h_{ij} and q^h_{ij} are bounded from above by the weight of a shortest odd and even walk, respectively, between vertices i and j in H with respect to edge variables w^h. Notice that every edge variable w^h_{jk} appears on the right hand side of inequalities (4.27) and (4.28), which allows p^h_{ij} and q^h_{ij} to attain the highest possible value if the respective f in the definition of w^h_{jk} attain the values of shortest odd walks in G. Inequalities (4.29) are satisfied by $(\bar{x}, \bar{f}, \bar{g}, \bar{p}, \bar{q})$ for all $i \in V$ and for every hub $h \in V$. Hence, the weight of a shortest odd cycle C in H has weight $\bar{w}^h(E_C)$ at least $1 - \bar{x}_h$ for every $h \in V$. By Theorem 3.7, no $(I_\mathcal{A})$-inequality is violated by $\bar{x}$ and finally $\bar{x} \in P^{W_\mathcal{A}}(G)$. □

Remark 4.17. *The extended formulation* $Q^{W_\mathcal{A}}(G)$ *requires* $n + 2n^2 + 2n^3 - n - n^2 = 2n^3 + n^2$ *variables. Notice that* w^h *can be replaced by their definition including the variables* x *and* f*. The number of inequalities is* $n + 2mn + 2mn + n + n^2 + n^4 + n^4 + n^2 = 2n^4 + 4mn + 2n^2 + 2n$.

Chapter 5

Compact extended relaxations for the BoxQP

The $\mathcal{NP}$-hard nonconvex quadratic program with box constraints (BoxQP) is a computationally challenging optimization problem. A very weak relaxation of the BoxQP is given by the McCormick relaxation. However, it can be strengthened with Chvátal-Gomory cuts for the Boolean Quadric Polytope (BQP) to obtain a very strong relaxation. All Chvátal-Gomory cuts for the BQP are dominated by the so-called A-odd cycle inequalities, which form an exponential class of inequalities in general.

This chapter covers the results from our manuscript *Tight compact extended relaxations for nonconvex quadratic programming problems with box constraints*, see de Vries and Perscheid (2019). We present new compact extended relaxations for the A-odd cycle inequalities of the BoxQP. This permits to optimize over relaxations that contain the A-odd cycle inequalities without being reliant on repeated calls to separation algorithms. In a computational study, we verify the strength of our new compact extended relaxations and show that we can significantly reduce the optimality gap of the BoxQP, even with a plain linear program.

5.1 The BoxQP

Bonami et al. (2018) show how the nonconvex quadratic programming problem with box constraints, which is defined as

$$\begin{aligned} \min \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \quad & l \leq x \leq u \\ & Q \in \mathbb{R}^{n \times n},\ Q \text{ symmetric}, \end{aligned} \tag{BoxQP}$$

can be solved effectively via linear programming techniques. Without loss of generality we

assume $l = \mathbf{0}$ and $u = \mathbf{1}$, because l and u are finite.

Let N denote the set $\{1, \ldots, n\}$ and

$$E \coloneqq \big\{(i,j) \in N \times N : i \neq j,\ Q_{ij} \neq 0\big\}.$$

Notice that if the ordered pair (i,j) is in $E \subseteq N \times N$, then $(j,i) \in E$ because of the symmetry of Q.

Bonami et al. first obtain a convex relaxation of the BoxQP with a linear objective function. This linearization induces nonlinear constraints, which are replaced by the so-called McCormick inequalities, see McCormick (1976). The resulting *weak* linear relaxation of the BoxQP is given by

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} Q_{ij}X_{ij} + \frac{1}{2}\sum_{i\in N} Q_{ii}Y_i + \sum_{i\in N} c_i x_i & & \\
\text{s.t.} \quad & x_i \geq Y_i \geq 2x_i - 1 & \forall\ i \in N & \\
& Y_i \geq 0 & \forall\ i \in N & \qquad (\mathrm{LP}_{\mathcal{M}}) \\
& x_i \geq X_{ij} & \forall\ (i,j) \in E & \\
& x_j \geq X_{ij} \geq x_i + x_j - 1 & \forall\ (i,j) \in E & \\
& X_{ij} \geq 0 & \forall\ (i,j) \in E. &
\end{aligned}$$

Furthermore, it can be strengthened to

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} Q_{ij}X_{ij} + \frac{1}{2}\sum_{i\in N^-} Q_{ii}Y_i \qquad + \frac{1}{2}\sum_{i\in N^+} Q_{ii}x_i^2 + \sum_{i\in N} c_i x_i & & \\
\text{s.t.} \quad & x_i \geq Y_i & \forall\ i \in N^- & \\
& x_i \geq X_{ij} & \forall\ (i,j) \in E & \qquad (\mathrm{QP}_{\mathcal{M}^2}) \\
& x_j \geq X_{ij} \geq x_i + x_j - 1 & \forall\ (i,j) \in E & \\
& X_{ij} \geq 0 & \forall\ (i,j) \in E & \\
& 1 \geq x_i \geq 0 & \forall\ i \in N, &
\end{aligned}$$

where $N^+ \coloneqq \{i \in N : Q_{ii} \geq 0\}$ and $N^- \coloneqq \{i \in N : Q_{ii} < 0\}$ partition N. Although $\mathrm{QP}_{\mathcal{M}^2}$ has a convex quadratic objective and only linear constraints apart from that, solving the pure $\mathrm{LP}_{\mathcal{M}}$ is much faster in practice. However, $\mathrm{QP}_{\mathcal{M}^2}$ provides better lower bounds for the BoxQP.

Cutting planes from the Boolean Quadric Polytope

$$\mathrm{BQP} = \mathrm{conv}\left(\mathrm{BQP}^{LP} \cap \left(\mathbb{Z}^n \times \mathbb{Z}^{|E|}\right)\right),$$

where

$$\mathrm{BQP}^{LP} = \Big\{(x, X) \in \mathbb{R}^n \times \mathbb{R}^{|E|} : \min\{x_i, x_j\} \geq X_{ij} \geq \max\{0, x_i + x_j - 1\} \quad \forall\ (i,j) \in E\Big\},$$

can be used to turn $\mathrm{LP}_{\mathcal{M}}$ and $\mathrm{QP}_{\mathcal{M}^2}$, respectively, into a very *strong* relaxation of the BoxQP. In particular, these efficient cuts are Chvátal-Gomory cuts

$$\alpha^T A x \geq \lceil \alpha^T b \rceil, \quad \alpha \in \mathbb{R}^m_+.$$

Moreover, all Chvátal-Gomory cuts for the BQP are $0 - \frac{1}{2}-$Chvátal-Gomory cuts (i.e., $\alpha \in \{0, \frac{1}{2}\}^m$), see Bonami et al. (2018). Caprara and Fischetti (1996) show that separating $0-\frac{1}{2}-$Chvátal-Gomory cuts is $\mathcal{NP}$-hard in general. However, Koster et al. (2009) study ways to separate them effectively in practice. Fortunately for our purposes, all $0 - \frac{1}{2}-$Chvátal-Gomory cuts of the BQP can be separated in polynomial time, as they are all dominated by the A-odd cycle inequalities, which is also proven by Bonami et al. (2018).

Boros et al. (1992) provide an extended formulation for the A-odd cycle inequalities for the BQP by adding $\mathcal{O}(n^2)$ variables and $\mathcal{O}(n^3)$ inequalities. We present an extended formulation in Section 5.3 which has also $\mathcal{O}(n^2)$ variables and just $\mathcal{O}(|E|n)$ inequalities. It is much more compact for sparse matrices Q, since we have $|E| \ll n^2$ in this case.

5.2 A-odd cycle inequalities for the BQP

The McCormick inequalities, cf. McCormick (1976), for the BQP^{LP} imply

$$0 \leq x_i \leq 1 \quad \forall\ i \in N,$$

since $0 \leq X_{ij} \leq x_i$ and $x_i + x_j - 1 \leq x_j$ for all $(i,j) \in E$. If we combine the McCormick inequalities $X_{ij} \geq 0$ and $X_{ij} \geq x_i + x_j - 1$, we have

$$2X_{ij} - x_i - x_j + 1 \geq 0. \tag{A_{ij}}$$

Analogously, adding up $x_i \geq X_{ij}$ and $x_j \geq X_{ij}$ yields

$$-2X_{ij} + x_i + x_j \geq 0. \tag{B_{ij}}$$

To obtain additional cuts for the BQP, combinations of (A_{ij})- and (B_{ij})-inequalities can be useful. Let $E^A \subseteq E$ be the set of all (i,j) for which we use inequality (A_{ij}). Define the set $\hat{E}^A$ as the set that contains an edge ij for every $(i,j) \in E^A$. The sets E^B and $\hat{E}^B$ are defined analogously. We combine (A_{ij})- and (B_{ij})-inequalities such that $|E^A|$ is odd and $\hat{E}^A \dot\cup \hat{E}^B$ is a cycle.

Definition 5.1. *A cycle $\hat{E}^A \dot\cup \hat{E}^B$ with $E^A, E^B \subseteq E$ is called* A-odd cycle *if $|E^A|$ is odd.*

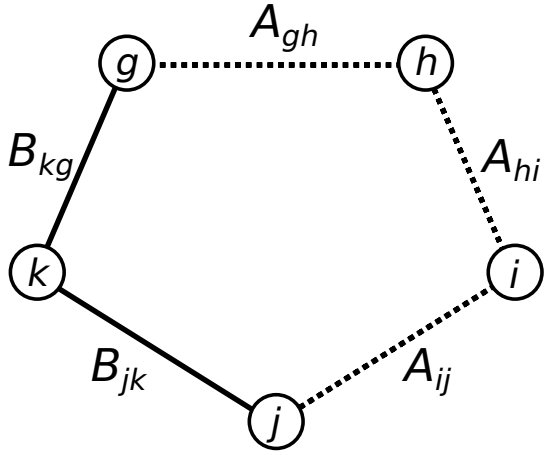


Figure 5.1: An A-odd cycle with $|E^A| = 3$.

In contrast to the odd cycle inequalities of the stable set polytope, the cycle $\hat{E}^A \dot{\cup} \hat{E}^B$ may be of arbitrary parity. Let

$$N^A \subseteq N: \text{ vertices incident to exactly two edges in } \hat{E}^A,$$
$$N^B \subseteq N: \text{ vertices incident to exactly two edges in } \hat{E}^B.$$

Remark 5.2. *Adding inequality (A_{ij}) to (B_{jk}) eliminates variable x_j.*

Example 5.3. *Let $(g,h),(h,i),(i,j),(j,k),(k,g) \in E$. If we add up (A_{gh}), (A_{hi}), (A_{ij}), (B_{jk}), and (B_{kg}), see Figure 5.1, we get*

$$2(X_{gh} + X_{hi} + X_{ij} - X_{jk} - X_{kg}) - 2(x_h + x_i) + 2x_k + 3 \geq 0.$$

Subtracting 3 and dividing by 2 yields

$$X_{gh} + X_{hi} + X_{ij} - X_{jk} - X_{kg} - x_h - x_i + x_k \geq -\frac{3}{2}$$

and as all variables on the left hand side are integral, we are able to round up the fractional constant on the right hand side to

$$X_{gh} + X_{hi} + X_{ij} - X_{jk} - X_{kg} - x_h - x_i + x_k \geq -1.$$

In general, adding up inequalities for a cycle $\hat{E}^A \dot{\cup} \hat{E}^B$ yields

$$2\left(\sum_{(i,j)\in E^A} X_{ij} - \sum_{(i,j)\in E^B} X_{ij} - \sum_{i\in N^A} x_i + \sum_{i\in N^B} x_i\right) + |E^A| \geq 0.$$

Subtracting $|E^A|$ and dividing by 2 yields

$$\sum_{(i,j)\in E^A} X_{ij} - \sum_{(i,j)\in E^B} X_{ij} - \sum_{i\in N^A} x_i + \sum_{i\in N^B} x_i \geq -\frac{|E^A|}{2}.$$

In the case $|E^A|$ is odd, we can strengthen this inequality, cf. Bonami et al. (2018), to

$$\sum_{(i,j)\in E^A} X_{ij} - \sum_{(i,j)\in E^B} X_{ij} - \sum_{i\in N^A} x_i + \sum_{i\in N^B} x_i \geq \left\lceil -\frac{|E^A|}{2} \right\rceil,$$

which is equivalent to

$$\sum_{(i,j)\in E^A} X_{ij} - \sum_{(i,j)\in E^B} X_{ij} - \sum_{i\in N^A} x_i + \sum_{i\in N^B} x_i \geq -\frac{|E^A|}{2} + \frac{1}{2},$$

and yields after another transformation

$$2\left(\sum_{(i,j)\in E^A} \left(X_{ij} + \tfrac{1}{2}\right) - \sum_{(i,j)\in E^B} X_{ij} - \sum_{i\in N^A} x_i + \sum_{i\in N^B} x_i\right) \geq 1. \tag{5.1}$$

Definition 5.4. *Inequalities* (5.1) *are called* A-odd cycle inequalities.

5.3 Separation & extended formulation

For inequalities (A_{ij}) and (B_{ij}), we define

$$w_{ij}^A = \quad 2X_{ij} - x_i - x_j + 1, \tag{5.2}$$
$$w_{ij}^B = -2X_{ij} + x_i + x_j. \tag{5.3}$$

Notice that w_{ij}^A and w_{ij}^B do not depend on some particular $\bar{x}$ or $\bar{X}$. They are variables restricted by the given equations and obviously nonnegative. We can interpret them as the slack of inequalities (A_{ij}) and (B_{ij}).

The following construction is similar to the separation algorithm for the cut polytope presented by Barahona and Mahjoub (1986). Nevertheless, we give a detailed explanation on how the separation problem for the A-odd cycle inequalities of the BQP can be solved.

Let $G = (V_G, E_G)$ be the simple graph with $V_G = N$ and $E_G = \{ij \ : \ (i,j),(j,i) \in E\}$. Consider the digraph $F = (V_F, A_F)$ with vertex set $V_F = \{0,1\}$ and arc set $A_F = \{(0,0),(0,1),(1,0),(1,1)\}$ as given in Figure 5.2. The categorical graph product $H = (V_{G\times F}, A_{G\times F})$ of G and F is given by the vertex set $V_{G\times F} = N \times \{0,1\}$ and the arc set $A_{G\times F} = \{((i,r),(j,s)) : ij \in E_G \text{ and } (r,s) \in A_F\}$.

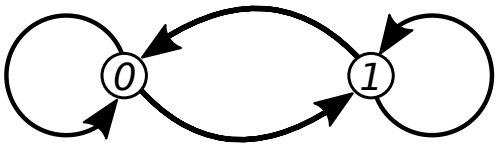


Figure 5.2: Digraph F.

Now we assign arc variable w_{ij}^A to the arcs $((i,r),(j,1-r)) \in A_{G\times F}$ and w_{ij}^B to the arcs $((i,r),(j,r)) \in A_{G\times F}$ for all $r \in \{0,1\}$. Figure 5.3 shows the structure of H for a given edge $ij \in E_G$.

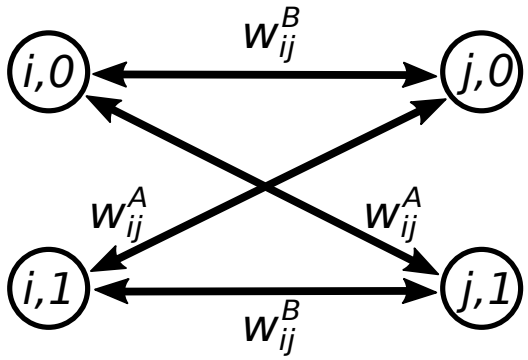


Figure 5.3: The categorical graph product of an edge ij and F. Antiparallel arcs are graphically represented by double-headed arrows.

Whenever we use an (A_{ij})-inequality for an edge $ij \in E_G$, an arc with arc variable w_{ij}^A in the product graph H is used and the second index of a vertex in H changes from 0 to 1 or from 1 to 0. Otherwise, using a (B_{ij})-inequality for an edge $ij \in E_G$ corresponds to an arc with arc variable w_{ij}^B in H and the second index does not change. We call a walk and a path, respectively, *A-odd* if the number of arcs with assigned arc variable w_{ij}^A is odd.

Lemma 5.5. *Every (u,r)-(v,s)-walk in H corresponds to an A-odd u-v-walk in G if and only if $r \neq s$.*

Proof. Let $r,s \in \{0,1\}$ with $r \neq s$. Then for every (u,r)-(v,s)-walk in H, the sum over all arc variables includes an odd number of variables w_{ij}^A since the second index does not change when using variables w_{ij}^B. The construction of variables w_{ij}^A that relate to inequalities (A_{ij}) yields that the corresponding u-v-walk in G is A-odd. Analogously, the corresponding u-v-walk in G is A-even if $r = s$. □

Lemma 5.6. *Let $(u,r) \neq (v,s)$. If a shortest (u,r)-(v,s)-walk in H has weight l, then there exists a (u,r)-(v,s)-*path *in H of weight l.*

Proof. Let P be a shortest A-odd (u,r)-(v,s)-walk in H with weight l. If P is a path, then there is nothing to show. Otherwise, there exists a closed walk through some vertex (w,t) in P with weight ≤ 0. Since all the edge variables in H are nonnegative, this closed walk has weight 0. Notice that it is A-even, since for every walk starting and ending in vertex (w,t), the second index alternates an even number of times between 0 and 1. Therefore, removing this walk from P does not change l or the parity of E^A. Removing all closed walks from P yields an A-odd (u,r)-(v,s)-path in H of weight l. □

Lemma 5.7. *The weight of a shortest A-odd cycle in G is equal to the weight of a shortest $(i,0)$-$(i,1)$-path in H among all $i \in N$.*

Proof. Notice first that the weight of a shortest $(i,0)$-$(i,1)$-path in H is equal to the weight of a shortest $(i,1)$-$(i,0)$-path in H because of arcs and arc weights being symmetric. Let P be a shortest $(i,0)$-$(i,1)$-path of all $(i,0)$-$(i,1)$-paths in H with $i \in N$. If the first index of all vertices except $(i,0)$ and $(i,1)$, which serve as start and end point, on P is different, then there is nothing to show. Otherwise, if for some j both vertices $(j,0)$ and $(j,1)$ lie on P, the subpath between $(j,0)$ and $(j,1)$ cannot have more weight than P as we do not have negative arc weights. Conversely, the subpath between $(j,0)$ and $(j,1)$ cannot have less weight than P by the assumption of P being one of the shortest of all $(i,0)$-$(i,1)$-paths in H with $i \in N$. Without loss of generality we can replace P by a shortest $(j,0)$-$(j,1)$-path. Successively, we end up in the first case. □

Lemma 5.8. *Given $(\bar{x},\bar{X}) \in BQP^{LP}$. Then $(\bar{x},\bar{X})$ violates an A-odd cycle inequality if and only if there exists a path P from $(i,0)$ to $(i,1)$ in H for some $i \in N$ of $\bar{w}$-weight less than 1.*

Proof. Let $\hat{E}^A \dot{\cup} \hat{E}^B$ be the edge set of a cycle C in G whose A-odd cycle inequality is violated by $(\bar{x},\bar{X})$. Then

$$2\left(\sum_{(i,j)\in E^A}\left(\bar{X}_{ij}+\tfrac{1}{2}\right)-\sum_{(i,j)\in E^B}\bar{X}_{ij}-\sum_{i\in N^A}\bar{x}_i+\sum_{i\in N^B}\bar{x}_i\right)<1.$$

by inequality (5.1). The left hand side can be transformed to

$$\sum_{(i,j)\in E^A}\left(2\bar{X}_{ij}+1-\bar{x}_i-\bar{x}_j\right)\ +\sum_{(i,j)\in E^B}\left(-2\bar{X}_{ij}+\bar{x}_i+\bar{x}_j\right),$$

since $\bar{x}_j$ is eliminated for all edge pairs ij and jk, where one of these edges appears in $\hat{E}^A$ and the other in $\hat{E}^B$, see Remark 5.2. This equals

$$\sum_{(i,j)\in E^A}\bar{w}^A_{ij}\ +\sum_{(i,j)\in E^B}\bar{w}^B_{ij}$$

and because $|E^A|$ is odd, there exist two paths from $(i,0)$ to $(i,1)$ and from $(i,1)$ to $(i,0)$

in H for all $i \in C$, that add up the same $\bar{w}^A_{ij}$ and $\bar{w}^B_{ij}$ as given above. Thus, the weight of each of these paths is less than 1.

For the converse, consider a path from $(i,0)$ to $(i,1)$ in H with weight less than 1. Analogously, there exists an A-odd cycle in G of equal weight. □

With Lemmas 5.5, 5.6, 5.7, and 5.8, we can state the following theorem:

Theorem 5.9. *For fixed $(\bar{x}, \bar{X})$, the separation problem for the A-odd cycle inequalities of the BQP can be solved by computing the weight of a shortest odd path from $(i,0)$ to $(i,1)$ in H for every $i \in N$. If every and hence the shortest of these paths has weight at least 1, then $(\bar{x}, \bar{X})$ does not violate any A-odd cycle inequality.*

Theorem 5.9 allows for solving the separation problem for the A-odd cycle inequalities of the BQP with a linear program. In Section 3.1.2, we have seen how to solve shortest path problems with a special case of the dual minimum cost flow problem (DMCF-SP). We apply this technique of Ahuja et al. (1993) and consider for fixed $i \in N$ the linear program

$$
\begin{aligned}
\max \quad & f_{i0i1} \\
\text{s.t.} \quad & f_{i0i0} = 0 \\
& f_{i0js} \leq f_{i0kt} + \bar{w}^A_{kj} && \forall\ (k,j) \in E,\ s,t \in \{0,1\},\ s \neq t \\
& f_{i0js} \leq f_{i0kt} + \bar{w}^B_{kj} && \forall\ (k,j) \in E,\ s,t \in \{0,1\},\ s = t
\end{aligned}
$$

with

$$
\begin{aligned}
\bar{w}^A_{kj} &= \ \ 2\bar{X}_{kj} - \bar{x}_k - \bar{x}_j + 1, \\
\bar{w}^B_{kj} &= -2\bar{X}_{kj} + \bar{x}_k + \bar{x}_j.
\end{aligned}
$$

Notice that we partition the inequalities into two types. Both of them bound the weight of a shortest $(i,0)$-(j,s)-path from above by the weight of any $(i,0)$-(j,s)-path where the last arc is fixed. The first type of inequalities ensures that the last arc on the path has arc weight $\bar{w}^A_{kj}$ whenever $s \neq t$ in order to arrive at vertex (j,s). Contrarily, we have to use arcs with arc weight $\bar{w}^B_{kj}$ whenever $s = t$.

If the objective value of a solution of this LP is greater or equal than 1 for every $i \in N$, then $(\bar{x}, \bar{X})$ fulfills all A-odd cycle inequalities of the BQP.

Of course, instead of solving n optimization problems, we can be formulate a feasibility problem with a similar technique as for the transformation from (DMCF-SP) to (F-SP) in Section 3.1.2. Using this idea, we obtain the following compact extended formulation that enforces all A-odd cycle inequalities. Notice that x and X as well as w are variables in contrast to what we have in the separation LP from above.

Theorem 5.10. *The linear system*

$$f_{irir} = 0 \qquad \forall\ i \in N,\ r \in \{0,1\} \tag{5.4}$$

$$f_{irjs} \leq f_{irkt} + w^A_{kj} \qquad \forall\ (k,j) \in E,\ i \in N,\ r,s,t \in \{0,1\},\ s \neq t \tag{5.5}$$

$$f_{irjs} \leq f_{irkt} + w^B_{kj} \qquad \forall\ (k,j) \in E,\ i \in N,\ r,s,t \in \{0,1\},\ s = t \tag{5.6}$$

$$f_{i0i1} \geq 1 \qquad \forall\ i \in N \tag{5.7}$$

together with equations (5.2) *and* (5.3) *is an extended formulation for the (potentially exponentially many) A-odd cycle inequalities of the BQP and therefore provides a relaxation for the BQP.*

Proof. Let $(\bar{x}, \bar{X}) \in \mathrm{BQP}^{LP}$. Then the weights $\bar{w}^A$ and $\bar{w}^B$ are explicitly given by equations (5.2) and (5.3).

We first show that if inequalities (5.1) are fulfilled by $(\bar{x}, \bar{X})$, then for every pair (i,r) and (j,s) in $V_{G\times F}$ there exists $\bar{f}_{irjs}$ such that $(\bar{x}, \bar{X}, \bar{f})$ is feasible for inequalities (5.4)–(5.7). Define $\bar{f}_{irjs}$ as the weight of a shortest (i,r)-(j,s)-path in H if such a path exists. Otherwise, assign a large value to $\bar{f}_{irjs}$. Inequalities (5.4) are obviously fulfilled, as shortest paths from a vertex to itself have weight 0 in digraphs where all arc weights are nonnegative. Inequalities (5.5) and (5.6) express that the weight of a shortest (i,r)-(j,s)-path cannot exceed the weight of an (i,r)-(j,s)-path where the last arc is fixed, which is always true. Finally, Lemma 5.8 ensures that inequalities (5.7) are fulfilled.

Conversely, let $(\bar{x}, \bar{X}, \bar{f})$ be feasible for inequalities (5.4)–(5.7). Then $\bar{f}_{irir} = 0$ for every $i \in N$ and $r \in \{0,1\}$ by equations (5.4), which is equal to the weight of a shortest path from vertex (i,r) in H to itself. Consider the case $(k,t) = (i,r)$ in inequalities (5.5) and (5.6). For every $(i,j) \in E$, variables f_{irjs} are bounded from above by arc variables w^A_{ij} if $r \neq s$. Moreover, variables f_{irjs} are bounded from above by arc variables w^B_{ij} if $r = s$. Taking those cases for inequalities (5.5) and (5.6) into account, where $(k,t) \neq (i,r)$, every variable f_{irjs} is bounded from above by the weight of a shortest path from (i,r) to (j,s) in H. Thus, for every vertex pair (i,r) and (j,s) in H, the value $\bar{f}_{irjs}$ is lower or equal than the weight of a shortest path from (i,r) to (j,s) in H. Since $\bar{f}_{i0i1}$ is lower or equal than the weight of a shortest $(i,0)$-$(i,1)$-path in H and $\bar{f}_{i0i1} \geq 1$ for $i \in N$, every shortest $(i,0)$-$(i,1)$-path in H has weight at least 1. This holds for every $i \in N$ and therefore all A-odd cycle inequalities (5.1) are fulfilled by $(\bar{x}, \bar{X})$, see Lemma 5.8. □

Remark 5.11. *Our extended A-odd cycle formulation in Theorem 5.10 requires* $4n^2$ *additional variables* f*, whereas the* w^A*- and* w^B*-defining equations can be replaced by their definition in terms of* x *and* X*. In total,* $8|E|n + n$ *inequalities are added. Notice that* f_{irir} *for all* $i \in N$ *and* $r \in \{0,1\}$ *are just constant numbers.*

5.4 Numerical experiments

For their computational study, Bonami et al. (2018) use CPLEX and add $0 - \frac{1}{2}-$Chvátal-Gomory cuts heuristically to the weak relaxations $\mathrm{LP}_{\mathcal{M}}$ and $\mathrm{QP}_{\mathcal{M}^2}$, respectively, from

Section 5.1. They apply this method for the 99 BoxQP test instances of Vandenbussche and Nemhauser (2005), Burer and Vandenbussche (2009), and Burer (2010). Our contribution is to compute the bounds that arise from *exact* A-odd cycle separation for the pure linear program $\mathrm{LP}_{\mathcal{M}}$. To this end, we add the extended relaxation from Theorem 5.10 to the constraint set of $\mathrm{LP}_{\mathcal{M}}$ and solve the resulting LP with CPLEX v. 12.8.0.0 on the same benchmark set. Adding the extended relaxation from Theorem 5.10 to $\mathrm{QP}_{\mathcal{M}^2}$ gives a convex quadratic program with a large amount of variables and inequalities. Although solving these QPs in reasonable running time does not seem to be promising, we compute the solutions for all instances with $n \leq 40$ whose density is not too high.

The optimality gap is defined as

$$\mathrm{gap}(z) \coloneqq \left| \frac{z_{\mathrm{BoxQP}} - z}{z} \right| \times 100,$$

where z_{BoxQP} is the optimal objective value of the BoxQP and z is the optimal objective value of the considered relaxation.

We denote the bounds arising from $\mathrm{LP}_{\mathcal{M}}$ and $\mathrm{QP}_{\mathcal{M}^2}$, respectively, by $z_{\mathcal{M}}$ and $z_{\mathcal{M}^2}$. For $\mathrm{LP}_{\mathcal{M}}$ strengthened with inequalities (5.2)–(5.7) we use the notation $\mathrm{LP}^{\bullet}_{\mathcal{M}}$. The bound arising from an optimal solution of $\mathrm{LP}^{\bullet}_{\mathcal{M}}$ is denoted by $z^{\bullet}_{\mathcal{M}}$. Analogously, we use the notation $z^{\bullet}_{\mathcal{M}^2}$ for the bound arising from an optimal solution of $\mathrm{QP}^{\bullet}_{\mathcal{M}^2}$, which is the extension of $\mathrm{QP}_{\mathcal{M}^2}$.

Table 5.1 is similar to Table 9 of Bonami et al. (2018), which also includes the columns z_{BoxQP}, $z_{\mathcal{M}}$, and $z_{\mathcal{M}^2}$. The bounds provided by the relaxations $\mathrm{LP}_{\mathcal{M}}$ and $\mathrm{QP}_{\mathcal{M}^2}$, respectively, where $0 - \frac{1}{2}-$Chvátal-Gomory cuts of the BQP are added heuristically, are replaced by the values $z^{\bullet}_{\mathcal{M}}$ and $z^{\bullet}_{\mathcal{M}^2}$ that arise from exact separation. Notice that the bounds $z^{\bullet}_{\mathcal{M}^2}$ are at least as strong as the bounds $z^{\bullet}_{\mathcal{M}}$. However, computing $z^{\bullet}_{\mathcal{M}^2}$ was only possible in reasonable time for instances with $n \in \{20, 30\}$ and for half of the instances with $n = 40$.

The first number in the name of the test instance is equal to n, i.e., the number of variables of the BoxQP. Moreover, the second number expresses the percentage of non-zeros in Q. The third number enumerates different test instances that have similar parameters.

Table 5.1: Bounds for BoxQP relaxations. All values in columns $z_{\mathcal{M}}$, $z_{\mathcal{M}^2}$, and z_{BoxQP} were taken from the computational study of Bonami et al. (2018).

Name	$z_{\mathcal{M}}$	$z_{\mathcal{M}^2}$	$z^{\bullet}_{\mathcal{M}}$	$z^{\bullet}_{\mathcal{M}^2}$	z_{BoxQP}
spar020-100-1	-1,066.00	-1,038.38	-706.50	-706.50	-706.50
spar020-100-2	-1,289.00	-1,258.38	-880.25	-867.14	-856.50
spar020-100-3	-1,168.50	-1,142.00	-772.00	-772.00	-772.00
spar030-060-1	-1,454.75	-1,430.00	-730.06	-714.21	-706.00
spar030-060-2	-1,699.50	-1,668.25	-1,385.50	-1,379.18	-1,377.17
spar030-060-3	-2,047.00	-2,006.50	-1,323.56	-1,305.97	-1,293.50
spar030-070-1	-1,569.00	-1,547.25	-703.86	-688.50	-654.00

spar030-070-2	-1,940.25	-1,888.25	-1,321.75	-1,315.82	-1,313.00
spar030-070-3	-2,302.75	-2,251.12	-1,695.00	-1,677.00	-1,657.40
spar030-080-1	-2,107.50	-2,072.00	-988.93	-967.73	-952.73
spar030-080-2	-2,178.25	-2,158.12	-1,597.00	-1,597.00	-1,597.00
spar030-080-3	-2,403.50	-2,376.25	-1,813.50	-1,809.78	-1,809.78
spar030-090-1	-2,423.50	-2,385.12	-1,296.50	-1,296.50	-1,296.50
spar030-090-2	-2,667.00	-2,622.75	-1,478.00	-1,470.64	-1,466.84
spar030-090-3	-2,538.25	-2,499.38	-1,494.00	-1,494.00	-1,494.00
spar030-100-1	-2,602.00	-2,541.50	-1,235.38	-1,227.38	-1,227.12
spar030-100-2	-2,729.25	-2,698.88	-1,260.50	-1,260.50	-1,260.50
spar030-100-3	-2,751.75	-2,703.75	-1,541.50	-1,524.07	-1,511.05
spar040-030-1	-1,088.00	-1,067.00	-839.50	-839.50	-839.50
spar040-030-2	-1,635.00	-1,617.75	-1,431.50	-1,429.36	-1,429.00
spar040-030-3	-1,303.25	-1,297.12	-1,086.00	-1,086.00	-1,086.00
spar040-040-1	-1,606.25	-1,575.50	-856.82	-847.93	-837.00
spar040-040-2	-1,920.75	-1,895.75	-1,428.00	-1,428.00	-1,428.00
spar040-040-3	-2,039.75	-2,017.25	-1,193.00	-1,179.26	-1,173.50
spar040-050-1	-2,146.25	-2,120.88	-1,157.00	-1,154.73	-1,154.50
spar040-050-2	-2,357.25	-2,334.88	-1,435.50	-1,432.04	-1,430.98
spar040-050-3	-2,616.00	-2,603.00	-1,658.00	-1,653.63	-1,653.63
spar040-060-1	-2,872.00	-2,817.88	-1,390.40	-1,365.00	-1,322.67
spar040-060-2	-2,917.50	-2,872.62	-2,014.00	-2,006.03	-2,004.23
spar040-060-3	-3,434.00	-3,386.12	-2,454.50	-2,454.50	-2,454.50
spar040-070-1	-3,144.00	-3,070.12	-1,605.00	–	-1,605.00
spar040-070-2	-3,369.25	-3,323.00	-1,867.50	–	-1,867.50
spar040-070-3	-3,760.25	-3,724.50	-2,444.00	–	-2,436.50
spar040-080-1	-3,846.50	-3,788.62	-1,838.50	–	-1,838.50
spar040-080-2	-3,833.00	-3,775.38	-1,952.50	–	-1,952.50
spar040-080-3	-4,361.50	-4,311.12	-2,561.50	–	-2,545.50
spar040-090-1	-4,376.75	-4,325.50	-2,135.50	–	-2,135.50
spar040-090-2	-4,357.75	-4,304.38	-2,123.29	–	-2,113.00
spar040-090-3	-4,516.75	-4,453.38	-2,540.00	–	-2,535.00
spar040-100-1	-5,009.75	-4,932.12	-2,487.50	–	-2,476.38
spar040-100-2	-4,902.75	-4,855.25	-2,146.25	–	-2,102.50
spar040-100-3	-5,075.75	-5,017.25	-2,192.17	–	-1,866.07
spar050-030-1	-1,858.25	-1,837.75	-1,324.50	–	-1,324.50
spar050-030-2	-2,334.00	-2,324.62	-1,669.00	–	-1,668.00
spar050-030-3	-2,107.25	-2,093.75	-1,461.00	–	-1,453.61

spar050-040-1	-2,632.00	-2,580.62	-1,411.00	–	-1,411.00
spar050-040-2	-2,923.25	-2,891.88	-1,753.50	–	-1,745.76
spar050-040-3	-3,273.50	-3,236.00	-2,094.50	–	-2,094.50
spar050-050-1	-3,536.00	-3,506.25	-1,409.72	–	-1,198.41
spar050-050-2	-3,500.50	-3,467.12	-1,776.81	–	-1,776.00
spar050-050-3	-4,119.75	-4,052.12	-2,138.34	–	-2,106.10
spar060-020-1	-1,757.25	-1,745.50	-1,212.00	–	-1,212.00
spar060-020-2	-2,238.25	-2,230.00	-1,925.50	–	-1,925.50
spar060-020-3	-2,098.75	-2,081.00	-1,483.00	–	-1,483.00
spar070-025-1	-3,832.75	-3,788.88	-2,545.00	–	-2,538.91
spar070-025-2	-3,248.00	-3,232.88	-1,888.50	–	-1,888.00
spar070-025-3	-4,167.25	-4,148.38	-2,819.25	–	-2,812.28
spar070-050-1	-7,210.75	-7,151.12	-3,356.00	–	-3,252.50
spar070-050-2	-6,620.00	-6,573.88	-3,296.00	–	-3,296.00
spar070-050-3	-7,522.00	-7,473.88	-4,306.50	–	-4,306.50
spar070-075-1	-11,647.75	-11,578.12	-5,003.67	–	-4,655.50
spar070-075-2	-10,884.75	-10,793.38	-4,504.92	–	-3,865.15
spar070-075-3	-11,262.25	-11,162.38	-4,862.75	–	-4,329.40
spar080-025-1	-4,840.75	-4,829.12	-3,157.00	–	-3,157.00
spar080-025-2	-4,378.50	-4,351.00	-2,361.62	–	-2,312.34
spar080-025-3	-5,130.25	-5,102.88	-3,101.00	–	-3,090.88
spar080-050-1	-9,783.25	-9,696.62	-4,025.80	–	-3,448.10
spar080-050-2	-9,270.00	-9,205.50	-4,450.50	–	-4,449.20
spar080-050-3	-10,029.75	-9,967.25	-4,961.27	–	-4,886.00
spar080-075-1	-15,250.75	-15,154.75	-6,601.92	–	-5,896.00
spar080-075-2	-14,246.50	-14,146.62	-5,953.17	–	-5,341.00
spar080-075-3	-14,961.50	-14,860.88	-6,584.00	–	-5,980.50
spar090-025-1	-6,171.50	-6,135.25	-3,423.78	–	-3,372.50
spar090-025-2	-6,015.00	-5,978.38	-3,550.65	–	-3,500.29
spar090-025-3	-6,698.25	-6,681.88	-4,299.00	–	-4,299.00
spar090-050-1	-12,584.00	-12,522.38	-5,468.90	–	-5,152.00
spar090-050-2	-11,920.50	-11,851.38	-5,404.36	–	-5,386.50
spar090-050-3	-12,514.00	-12,452.50	-6,230.59	–	-6,151.00
spar090-075-1	-19,054.25	-18,944.50	-7,944.92	–	-6,267.45
spar090-075-2	-18,245.50	-18,132.50	-7,334.75	–	-5,647.50
spar090-075-3	-18,929.50	-18,823.50	-7,908.50	–	-6,450.00
spar100-025-1	-7,660.75	-7,611.38	-4,116.48	–	-4,027.50
spar100-025-2	-7,338.50	-7,303.12	-3,906.07	–	-3,892.56

spar100-025-3	-7,942.25	-7,894.75	-4,459.25	–	-4,453.50
spar100-050-1	-15,415.75	-15,341.75	-6,366.84	–	-5,490.00
spar100-050-2	-14,920.50	-14,814.62	-6,504.93	–	-5,866.00
spar100-050-3	-15,564.25	-15,480.12	-7,031.72	–	-6,485.00
spar100-075-1	-23,387.50	-23,277.12	-9,551.75	–	-7,384.20
spar100-075-2	-22,440.00	-22,307.00	-8,826.42	–	-6,755.50
spar100-075-3	-23,243.50	-23,109.62	-9,614.25	–	-7,554.00
spar125-025-1	-12,251.00	-12,184.75	-6,118.03	–	-5,572.00
spar125-025-2	-12,732.00	-12,662.62	-6,401.29	–	-6,156.06
spar125-025-3	-12,650.75	-12,627.50	-6,923.00	–	-6,815.50
spar125-050-1	-24,993.00	-24,880.25	-10,879.42	–	-9,308.38
spar125-050-2	-24,810.50	-24,669.38	-10,273.75	–	-8,395.00
spar125-050-3	-24,424.50	-24,308.00	-10,032.50	–	-8,343.91
spar125-075-1	-38,202.00	-38,058.12	-16,053.67	–	-12,330.00
spar125-075-2	-37,466.75	-37,341.38	-15,088.58	–	-10,382.47
spar125-075-3	-36,202.25	-36,033.00	-13,917.67	–	-9,635.50

In Table 5.2, we list the optimality gap for every relaxation $\mathrm{LP}_{\mathcal{M}}$, $\mathrm{QP}_{\mathcal{M}^2}$, $\mathrm{LP}^{\bullet}_{\mathcal{M}}$, and $\mathrm{QP}^{\bullet}_{\mathcal{M}^2}$ on every test instance, except for those $\mathrm{QP}^{\bullet}_{\mathcal{M}^2}$ that were not solved.

Table 5.2: Optimality gap for BoxQP relaxations.

Name	$\mathrm{gap}(z_{\mathcal{M}})$	$\mathrm{gap}(z_{\mathcal{M}^2})$	$\mathrm{gap}(z^{\bullet}_{\mathcal{M}})$	$\mathrm{gap}(z^{\bullet}_{\mathcal{M}^2})$
spar020-100-1	33.72	31.96	0.00	0.00
spar020-100-2	33.55	31.94	2.70	1.23
spar020-100-3	33.93	32.40	0.00	0.00
spar030-060-1	51.47	50.63	3.30	1.15
spar030-060-2	18.97	17.45	0.60	0.15
spar030-060-3	36.81	35.53	2.27	0.95
spar030-070-1	58.32	57.73	7.08	5.01
spar030-070-2	32.33	30.46	0.66	0.21
spar030-070-3	28.03	26.37	2.22	1.17
spar030-080-1	54.79	54.02	3.66	1.55
spar030-080-2	26.68	26.00	0.00	0.00
spar030-080-3	24.70	23.84	0.21	0.00
spar030-090-1	46.50	45.64	0.00	0.00
spar030-090-2	45.00	44.07	0.76	0.26
spar030-090-3	41.14	40.23	0.00	0.00

spar030-100-1	52.84	51.72	0.67	0.02
spar030-100-2	53.82	53.30	0.00	0.00
spar030-100-3	45.09	44.11	1.98	0.85
spar040-030-1	22.84	21.32	0.00	0.00
spar040-030-2	12.60	11.67	0.17	0.03
spar040-030-3	16.67	16.28	0.00	0.00
spar040-040-1	47.89	46.87	2.31	1.29
spar040-040-2	25.65	24.67	0.00	0.00
spar040-040-3	42.47	41.83	1.63	0.49
spar040-050-1	46.21	45.57	0.22	0.02
spar040-050-2	39.29	38.71	0.31	0.07
spar040-050-3	36.79	36.47	0.26	0.00
spar040-060-1	53.95	53.06	4.87	3.10
spar040-060-2	31.30	30.23	0.49	0.09
spar040-060-3	28.52	27.51	0.00	0.00
spar040-070-1	48.95	47.72	0.00	–
spar040-070-2	44.57	43.80	0.00	–
spar040-070-3	35.20	34.58	0.31	–
spar040-080-1	52.20	51.47	0.00	–
spar040-080-2	49.06	48.28	0.00	–
spar040-080-3	41.64	40.96	0.62	–
spar040-090-1	51.21	50.63	0.00	–
spar040-090-2	51.51	50.91	0.48	–
spar040-090-3	43.88	43.08	0.20	–
spar040-100-1	50.57	49.79	0.45	–
spar040-100-2	57.12	56.70	2.04	–
spar040-100-3	63.24	62.81	14.88	–
spar050-030-1	28.72	27.93	0.00	–
spar050-030-2	28.53	28.25	0.06	–
spar050-030-3	31.02	30.57	0.51	–
spar050-040-1	46.39	45.32	0.00	–
spar050-040-2	40.28	39.63	0.44	–
spar050-040-3	36.02	35.28	0.00	–
spar050-050-1	66.11	65.82	14.99	–
spar050-050-2	49.26	48.78	0.05	–
spar050-050-3	48.88	48.02	1.51	–
spar060-020-1	31.03	30.56	0.00	–
spar060-020-2	13.97	13.65	0.00	–

spar060-020-3	29.34	28.74	0.00	–
spar070-025-1	33.76	32.99	0.24	–
spar070-025-2	41.87	41.60	0.03	–
spar070-025-3	32.51	32.21	0.25	–
spar070-050-1	54.89	54.52	3.08	–
spar070-050-2	50.21	49.86	0.00	–
spar070-050-3	42.75	42.38	0.00	–
spar070-075-1	60.03	59.79	6.96	–
spar070-075-2	64.49	64.19	14.20	–
spar070-075-3	61.56	61.21	10.97	–
spar080-025-1	34.78	34.63	0.00	–
spar080-025-2	47.19	46.85	2.09	–
spar080-025-3	39.75	39.43	0.33	–
spar080-050-1	64.76	64.44	14.35	–
spar080-050-2	52.00	51.67	0.03	–
spar080-050-3	51.28	50.98	1.52	–
spar080-075-1	61.34	61.09	10.69	–
spar080-075-2	62.51	62.25	10.28	–
spar080-075-3	60.03	59.76	9.17	–
spar090-025-1	45.35	45.03	1.50	–
spar090-025-2	41.81	41.45	1.42	–
spar090-025-3	35.82	35.66	0.00	–
spar090-050-1	59.06	58.86	5.79	–
spar090-050-2	54.81	54.55	0.33	–
spar090-050-3	50.85	50.60	1.28	–
spar090-075-1	67.11	66.92	21.11	–
spar090-075-2	69.05	68.85	23.00	–
spar090-075-3	65.93	65.73	18.44	–
spar100-025-1	47.43	47.09	2.16	–
spar100-025-2	46.96	46.70	0.35	–
spar100-025-3	43.93	43.59	0.13	–
spar100-050-1	64.39	64.22	13.77	–
spar100-050-2	60.68	60.40	9.82	–
spar100-050-3	58.33	58.11	7.78	–
spar100-075-1	68.43	68.28	22.69	–
spar100-075-2	69.90	69.72	23.46	–
spar100-075-3	67.50	67.31	21.43	–
spar125-025-1	54.52	54.27	8.92	–

spar125-025-2	51.65	51.38	3.83	–
spar125-025-3	46.13	46.03	1.55	–
spar125-050-1	62.76	62.59	14.44	–
spar125-050-2	66.16	65.97	18.29	–
spar125-050-3	65.84	65.67	16.83	–
spar125-075-1	67.72	67.60	23.20	–
spar125-075-2	72.29	72.20	31.19	–
spar125-075-3	73.38	73.26	30.77	–

Let d be the percentage of non-zeros in Q. An instance is called *sparse*, *medium*, or *dense*, if $d \leq 40\%$, $40\% < d \leq 60\%$, or $d > 60\%$, respectively. Moreover, we divide these classes further into *small* ($n \in \{20, 30, 40\}$), *medium* ($n \in \{50, 60, 70\}$), *large* ($n \in \{80, 90\}$), and *jumbo* ($n \in \{100, 125\}$).

The set of small test instances is partitioned into 6 sparse, 9 medium dense, and 27 dense instances. Table 5.3 specifies how much of the optimality gap is closed by the A-odd cycle inequalities when adding them to $\mathrm{LP}^{\bullet}_{\mathcal{M}}$ and $\mathrm{QP}^{\bullet}_{\mathcal{M}^2}$, respectively.

Density	$\mathrm{gap}(z_{\mathcal{M}})$	$\mathrm{gap}(z_{\mathcal{M}^2})$	$\mathrm{gap}(z^{\bullet}_{\mathcal{M}})$	$\mathrm{gap}(z^{\bullet}_{\mathcal{M}^2})$
Sparse	28.02	27.11	0.69	0.30
Medium	38.15	37.24	1.37	0.61
Dense	44.43	43.50	1.44	–

Table 5.3: Average optimality gap for instances with $n \in \{20, 30, 40\}$. Time limit is exceeded when solving $\mathrm{QP}^{\bullet}_{\mathcal{M}^2}$ for dense instances with $n = 40$.

The average optimality gap of $\mathrm{LP}_{\mathcal{M}}$, $\mathrm{QP}_{\mathcal{M}^2}$, and $\mathrm{LP}^{\bullet}_{\mathcal{M}}$, respectively, for all medium, large, and jumbo instances is visualized graphically in Figure 5.4.

We obtain that the pure linear relaxation $\mathrm{LP}_{\mathcal{M}}$ together with the A-odd cycle inequalities closes by far more of the optimality gap than the quadratic relaxation $\mathrm{QP}_{\mathcal{M}^2}$ without the A-odd cycle inequalities, regardless of the choice of the size n or the density d. Moreover, the impact of the A-odd cycle inequalities increases when decreasing the density of Q. Especially on sparse instances, the optimality gap is reduced tremendously by using the relaxation $\mathrm{LP}^{\bullet}_{\mathcal{M}}$.

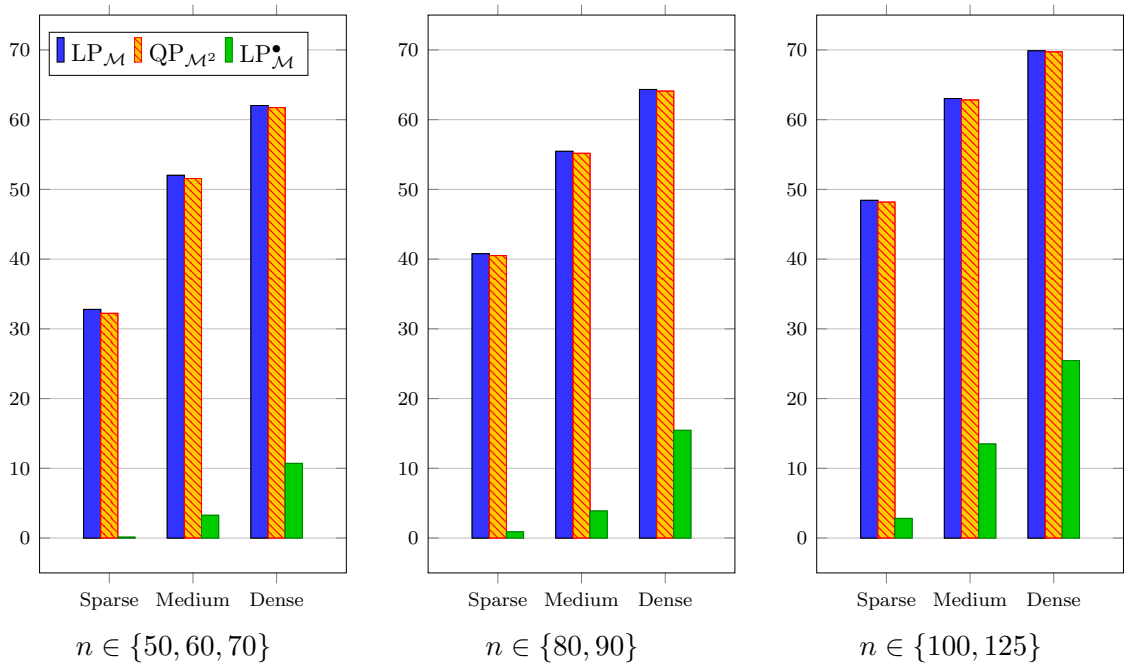


Figure 5.4: Average optimality gap by category.

Chapter 6

Integer solutions for the p-median problem

Let $D = (V, A)$ be a digraph, where every arc $(i, j) \in A$ has an associated cost c_{ij}. The p-median problem is to select p vertices and assign every of the $n - p$ remaining vertices $i \in V$ to a selected vertex j via an arc $(i, j) \in A$ such that the sum of costs yielded by the selection and the assignment of nonselected vertices is minimized. This problem is $\mathcal{NP}$-hard, even if the network has a simple structure, e.g. it is planar or the maximum vertex degree is 3, see Kariv and Hakimi (1979). Furthermore, Baïou et al. (2013) prove that even under the strong restriction that D is triangle-free (that is D does not contain any directed 3-cycle as a subgraph), it remains $\mathcal{NP}$-hard in general. The p-median problem was extensively studied since the early 1960's and is related to the well-known *uncapacitated facility location problem*. Reese (2006) summarizes literature on solution methods for the p-median problem that include several heuristics, enumeration methods, LP relaxations, and IP formulations.

The p-median polytope is the convex hull of integer points for the p-median problem. A linear relaxation of the p-median polytope that includes the directed odd cycle inequalities is given by Baïou and Barahona (2008). Moreover, they show that in the very special case of D being Y-free, which will be defined below, their relaxation is of polynomial size and it is sufficient to completely describe the p-median polytope. Baïou and Barahona (2016) extend their analysis of the effect of the directed odd cycle inequalities applied to the p-median problem by considering oriented digraphs. They specify subgraphs whose absence in D guarantees that the directed odd cycle relaxation is equal to the p-median polytope.

6.1 The p-median polytope

In this section, some important results on the p-median polytope, that are necessary for our extended formulation in Section 6.2, are summarized. All these results are cited from articles by Baïou and Barahona (2008, 2011, 2016). We mainly use their notation and only make small adjustments in order to achieve consistency throughout all chapters of this thesis.

The p-median problem can be formulated as the integer linear optimization problem

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{i\in V} y_i = p & & (6.1)\\
& y_i + \sum_{j:(i,j)\in A} x_{ij} = 1 & \forall\, i \in V & (6.2)\\
& x_{ij} \leq y_j & \forall\, (i,j) \in A & (6.3)\\
& y_i \in \{0,1\} & \forall\, i \in V & (6.4)\\
& x_{ij} \in \{0,1\} & \forall\, (i,j) \in A, & (6.5)
\end{aligned}$$

see Baïou and Barahona (2008). In literature, we sometimes have additional costs caused by assigning positive values to the vertex variables. These costs $\sum_{i\in V} d_i y_i$, where d_i is the cost for selecting vertex $i \in V$, then have to be added to the objective function.

Remark 6.1. *An optimal solution for the p-median problem could be derived from an optimal solution of the system where the binary constraints* (6.5) *are replaced by* $0 \leq x_{ij} \leq 1$ *for every* $(i,j) \in A$*. An arc variable* x_{ij} *can only be greater than* 0 *if* $y_i = 0$ *and* $y_j = 1$*, respectively, by equations* (6.2) *and inequalities* (6.3) *in combination with constraints* (6.4)*. If the value of an arc variable* x_{ij} *is fractional, there must be at least one arc variable* x_{ik} *with* $k \neq j$ *which also takes a fractional value because of equations* (6.2)*. Moreover, we have* $c_{ij} = c_{ik}$*, since otherwise this would contradict optimality. Then we are able to set* $x_{ij} = 1$ *and* $x_{ik} = 0$ *for all arcs* $(i,k) \in A$ *with* $k \neq j$*. Applying this procedure until there is no fractional value for any variable* x *anymore yields an optimal solution for the p-median problem. Hence, any vertex of a relaxation for the p-median problem whose y-variables are integral is integral.*

However, as Baïou and Barahona, we work with the formulation from above that contains constraints (6.5) in this chapter.

As the convex hull of feasible points of the p-median problem, we obtain the so-called p-median polytope

$$p\,\mathrm{MP}(D) \coloneqq \mathrm{conv}\left\{(x,y) \in \mathbb{R}^m \times \mathbb{R}^n : (x,y) \text{ satisfies (6.1)–(6.5)}\right\}.$$

6.1.1 Relaxations of $p\,\mathrm{MP}(D)$

The natural linear relaxation for the p-median problem is denoted by $P_p(D)$ and given by replacing constraints (6.4) and (6.5), respectively, by $0 \leq y_i \leq 1$ for every $i \in V$ and $0 \leq x_{ij} \leq 1$ for every $(i,j) \in A$. However, it is not necessary to bound any variable x or y from above by 1, since the nonnegativity constraints of all variables x and y together with equations (6.2) imply $y_i \leq 1$ for every $i \in V$ and $x_{ij} \leq 1$ for every $(i,j) \in A$. Thus, we

define

$$P_p(D) \coloneqq \{(x,y) \in \mathbb{R}^m \times \mathbb{R}^n : (x,y) \text{ satisfies (6.1)–(6.3), (6.6), and (6.7)}\}$$

with

$$y_i \geq 0 \qquad \forall\, i \in V \tag{6.6}$$
$$x_{ij} \geq 0 \qquad \forall\, (i,j) \in A. \tag{6.7}$$

Baïou and Barahona (2008) show that the *(directed) odd cycle inequalities*

$$\sum_{(i,j)\in A_C} x_{ij} \leq \frac{|A_C|-1}{2} \qquad \forall \text{ directed odd cycles } C \tag{6.8}$$

are valid for $p\,\mathrm{MP}(D)$. This enables strengthening the relaxation $P_p(D)$, which is of polynomial size, by adding these inequalities to $P_p(D)$. We denote the resulting polytope by

$$P_p^{OC}(D) \coloneqq \{(x,y) \in \mathbb{R}^m \times \mathbb{R}^n : (x,y) \text{ satisfies (6.1)–(6.3) and (6.6)–(6.8)}\}.$$

As there may be exponentially many (directed) odd cycles in D, this formulation is not polynomial in general.

Before we present an extended formulation of $P_p^{OC}(D)$ in Section 6.2, we want to point out some conditions to D that are sufficient for $P_p(D)$ and $P_p^{OC}(D)$, respectively, to be integral. Then, the respective formulation equals the p-median polytope $p\,\mathrm{MP}(D)$.

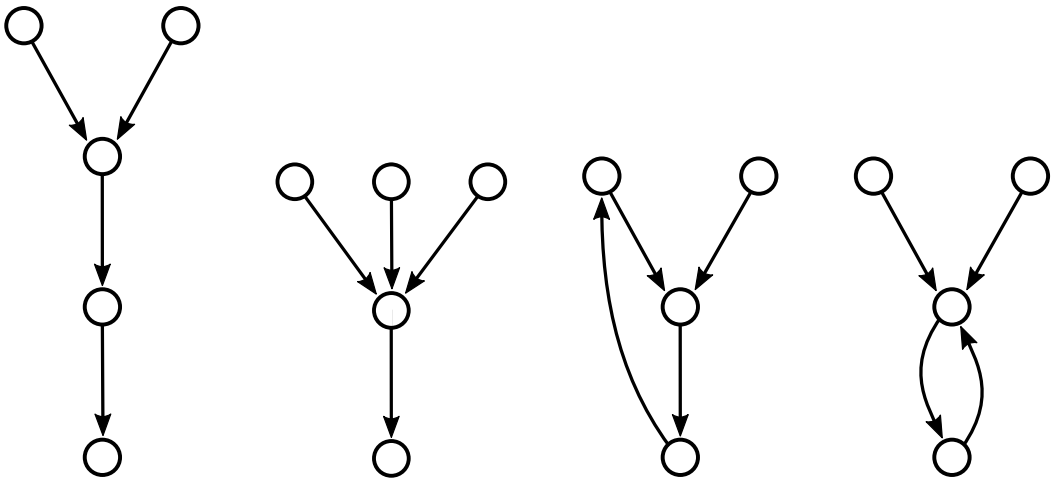
6.1.2 Integrality of $P_p(D)$

We now summarize some results of Baïou and Barahona (2011), who extensively analyze $P_p(D)$. They show that the absence of specific subgraphs of D is equivalent to $P_p(D)$ being integral. This allows us to classify digraphs for which we can solve the p-median problem in polynomial time with this simple formulation.

Some of the critical subgraphs for $P_p(D)$ are explicitly given in Figure 6.1. Fractional extreme points for the case that D is one of the digraphs H_1, H_2, H_3, or H_4, where $p = 3$ for H_1 and H_2, and $p = 2$ for H_3 and H_4, are highlighted in Figure 6.2. Every arc variable x takes the value $\frac{1}{2}$, which we illustrate graphically with dashed arcs. As usual, the proportion of red color within each vertex indicates the value of the corresponding variable y. In all of the four digraphs, the value of y_i is either $\frac{1}{2}$ or 1 for every vertex $i \in V$.

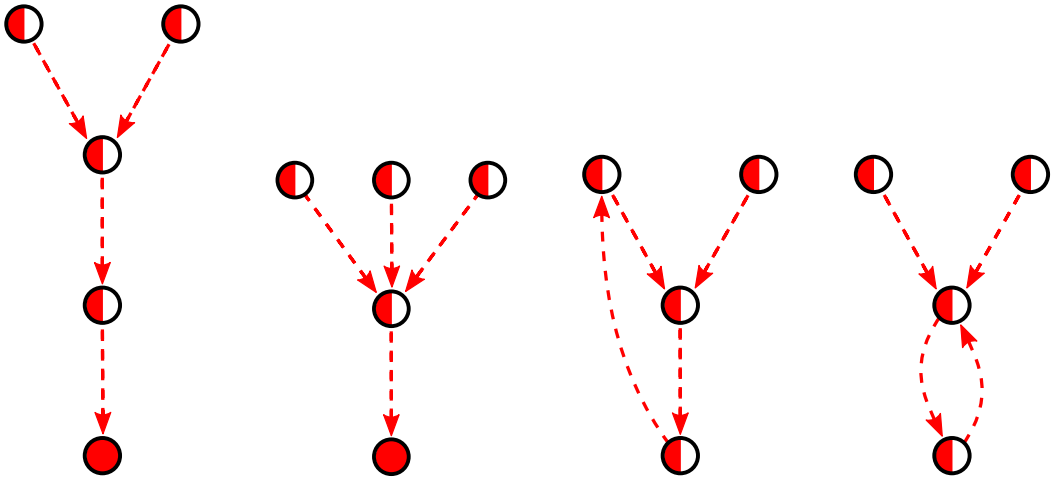
Another general class of subgraphs with a more complicated structure is very important for relaxations of the p-median polytope. This class is constituted by the so-called *weak Y-cycles*. These subgraphs are not isomorphic to a specific digraph as the four digraphs H_1, H_2, H_3, and H_4 are. They are extensions of *weak cycles*, which we are going to define first.

Definition 6.2. *Given a digraph $D = (V, A)$, we call a subgraph $\tilde{C}$ of D a* weak cycle *if replacing every arc $(u, v) \in \tilde{C}$ by an edge uv yields an undirected cycle. We partition $\tilde{C}$ into*



(a) H_1 (b) H_2 (c) H_3 (d) H_4

Figure 6.1: Critical subgraphs for relaxations of the p-median polytope.



(a) H_1 (b) H_2 (c) H_3 (d) H_4

Figure 6.2: Fractional extreme points of $P_p(D)$.

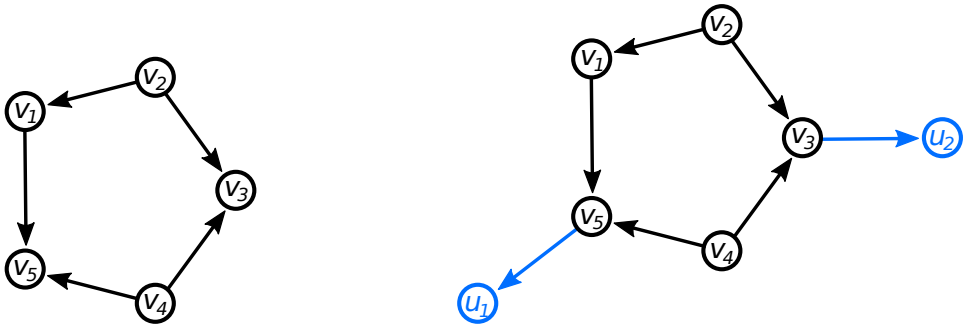
$\tilde{C}^0$, $\tilde{C}^1$, and $\tilde{C}^2$:

- *$\tilde{C}^0$ contains all vertices v which are the tail of two arcs in $\tilde{C}$,*
- *$\tilde{C}^1$ contains all vertices v which are the tail of one arc in $\tilde{C}$ and the head of another arc in $\tilde{C}$,*
- *$\tilde{C}^2$ contains all vertices v which are the head of two arcs in $\tilde{C}$.*

Moreover, a weak cycle is called g-odd *if $|\tilde{C}^1| + |\tilde{C}^2|$ is odd.*

By means of the vertex partition $\tilde{C} = \tilde{C}^0 \dot\cup \tilde{C}^1 \dot\cup \tilde{C}^2$ of a weak cycle $\tilde{C}$, we are able to give the following definition.

Definition 6.3. *A weak cycle $\tilde{C}$ is called a* weak Y-cycle *if for every $i \in \tilde{C}^2$ there is an arc $(i, j) \in A$ with $j \in V \setminus \tilde{C}^0$.*



(a) (b)

Figure 6.3: A g-odd weak cycle and a g-odd weak Y-cycle in D.

An example of a g-odd weak Y-cycle is given in Figure 6.3(b). It is the same g-odd weak cycle as in Figure 6.3(a), but we additionally have the information that there exist arcs (v_5, u_1) and (v_3, u_2) in D. Notice that $v_3, v_5 \in \tilde{C}^2$ and $u_1, u_2 \notin \tilde{C}^0 = \{v_2, v_4\}$.

Theorem 6.4. *(Baïou and Barahona, 2011, Theorem 2) For any digraph $D = (V, A)$, the polytope $P_p(D)$ is integral for any integer p if and only if*

- *D does not contain any of the digraphs H_1, H_2, H_3, nor H_4 as a subgraph, and*
- *D does not contain a g-odd weak Y-cycle $\tilde{C}$ and an arc (i, j) with $i, j \notin \tilde{C}$.*

6.1.3 Integrality of $P_p^{OC}(D)$

Some first observations regarding the relaxation $P_p^{OC}(D)$ were made by Baïou and Barahona (2008). They consider the restriction of D being Y-free and show that this implies $P_p^{OC}(D)$ being integral. Similarly, Baïou and Barahona (2016) analyze critical subgraphs of D for the general case. They prove that $P_p^{OC}(D)$ is integral if and only if none of those subgraphs is present in D. In the further course of this section, we summarize interesting results from both articles mentioned above.

Definition 6.5. *A digraph D is Y-*free *if D is oriented and it does not contain the subgraph Y of Figure 6.4.*

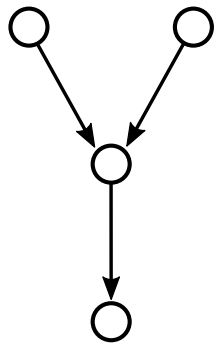


Figure 6.4: The subgraph Y.

The property of D being Y-free is a very strong restriction. However, this permits the following characterization.

Theorem 6.6. *(Baïou and Barahona, 2008, Theorem 20) If D is Y-free then $P_p^{OC}(D)$ is integral for all p.*

Remark 6.7. *The number of odd cycles in Y-free digraphs is polynomially bounded since each arc in those digraphs can belong to at most one cycle.*

Theorem 6.6 suggests that the converse does not hold, i.e. integrality of $P_p^{OC}(D)$ does not imply D is Y-free. Indeed, we can easily confirm this with a much stronger result. Theorem 6.8 provides equivalent conditions to $P_p^{OC}(D)$ being integral.

Theorem 6.8. *(Baïou and Barahona, 2016, Theorem 3) Let $D = (V, A)$ be triangle-free and oriented. The polytope $P_p^{OC}(D)$ is integral for any integer p if and only if*

- *D does not contain any of the digraphs H_1 nor H_2 as a subgraph, and*
- *D does not contain a nondirected g-odd weak Y-cycle $\tilde{C}$ and an arc (i, j) with $i, j \notin \tilde{C}$.*

For proving integrality of $P_p^{OC}(D)$ for triangle-free, oriented digraphs, only subgraphs H_1, H_2, and *nondirected* g-odd weak Y-cycles are important due to Theorem 6.8. The subgraphs H_3 and H_4, which are important for Theorem 6.4, cannot be present in D since H_3 is not triangle-free and H_4 is not oriented. The reason not to care about *directed* g-odd weak Y-cycles is that the polytope $P_p^{OC}(D)$ includes the odd cycle inequalities (6.8).

6.2 An extended formulation of $P_p^{OC}(D)$

Let $(\bar{x}, \bar{y}) \in P_p(D)$. For every odd cycle C in D, the odd cycle inequality

$$\sum_{(i,j)\in A_C} \bar{x}_{ij} \leq \frac{|A_C| - 1}{2}$$

of $P_p^{OC}(D)$ is equivalent to

$$1 \leq |A_C| - 2 \sum_{(i,j)\in A_C} \bar{x}_{ij} = \sum_{(i,j)\in A_C} (1 - 2\bar{x}_{ij}) = \sum_{j:(i,j),(j,k)\in A_C} (1 - \bar{x}_{ij} - \bar{x}_{jk}) =: \bar{w}(A_C).$$

Hence, every odd cycle C in D must have weight $\bar{w}(A_C)$ at least 1. This gives rise to two different separation algorithms for the odd cycle inequalities (6.8) of the p-median polytope, cf. Baïou and Barahona (2008).

The first one uses an auxiliary graph, which is the linegraph of D. It has a vertex (i, j) for every arc $(i, j) \in A$ and two vertices (i, j) and (k, l) are connected by an arc $((i, j), (k, l))$ if and only if $j = k$. Then we can define arc weights $\bar{w}_{ijk} := 1 - \bar{x}_{ij} - \bar{x}_{jk}$ for every arc $((i, j), (j, k))$ in the linegraph of D.

Lemma 6.9. *All arc weights $\bar{w}_{ijk}$ in the linegraph of D are nonnegative.*

Proof. Consider two consecutive arcs (i, j) and (j, k) in D. From equations (6.2) we obtain

$$\bar{y}_j = 1 - \sum_{l:(j,l)\in A} \bar{x}_{jl}$$

and together with $\bar{x}_{ij} \leq \bar{y}_j$ from inequalities (6.3) we have

$$\bar{x}_{ij} \leq 1 - \sum_{l:(j,l)\in A} \bar{x}_{jl},$$

which is equivalent to

$$0 \leq 1 - \bar{x}_{ij} - \bar{x}_{ik} - \sum_{\substack{l:(j,l)\in A \\ l\neq k}} \bar{x}_{jl}.$$

Hence $1 - \bar{x}_{ij} - \bar{x}_{jk} \geq 0$ because $\bar{x}_{jl} \geq 0$ for all $(j, l) \in A$. □

Thus, the separation algorithm for the odd cycle inequalities of the p-median polytope applied to the linegraph of D with weights $\bar{w}_{ijk}$ is equivalent to the separation algorithm for the odd cycle inequalities of the stable set polytope. Since the linegraph of D has a vertex for every arc in D, we would have to compute m shortest paths in the categorical graph product of the linegraph of D and $H = (V_H, E_H)$ with $V_H = \{0, 1\}$ and $E_H = \big\{\{0, 1\}\big\}$ from Section 3.2.

Baïou and Barahona (2008) give an alternative separation algorithm for the odd cycle inequalities of the p-median polytope, which uses arc weights $\bar{w}_{ij} \coloneqq 1 - 2\bar{x}_{ij}$ for every arc $(i, j) \in A$ and only requires computing n shortest paths in the categorical graph product of D and $H = (V_H, E_H)$ with $V_H = \{0, 1\}$ and $E_H = \big\{\{0, 1\}\big\}$. In contrast to the separation algorithm for the odd cycle inequalities of the stable set polytope, arc weights can be negative. Fortunately, this does not cause a problem.

Lemma 6.10. *(Baïou and Barahona, 2008, Lemma 2) The auxiliary graph $D \times H$ has no cycle with negative weight.*

The proof follows directly from Lemma 6.9 and the fact that the weight $\bar{w}(A_C)$ of every cycle C can be given as the sum of nonnegative arc weights.

We use this separation algorithm for the construction of an extended formulation $Q_p^{OC}(D)$ of the polytope $P_p^{OC}(D)$, which is closely related to $Q_2^{OC}(G)$ of the odd cycle polytope for the maximum stable set problem. Notice that giving an extended formulation by following the construction of $Q_1^{OC}(G)$ is not possible for our purposes here, since the arc weights $\bar{w}_{ij} = 1 - 2\bar{x}_{ij}$ in D can be negative.

We define

$$Q_p^{OC}(D) \coloneqq \big\{(x, y, f, g) \in \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} : (x, y, f, g) \text{ satisfies } (6.1)\text{–}(6.3),\ (6.6),\ (6.7),\ \text{and } (6.9)\text{–}(6.12)\big\}$$

with

$$g_{ii} = 0 \qquad \forall\ i \in V \tag{6.9}$$
$$f_{ij} \leq g_{ik} + 1 - 2x_{kj} \qquad \forall\ (k, j) \in A,\ i \in V \tag{6.10}$$
$$g_{ij} \leq f_{ik} + 1 - 2x_{kj} \qquad \forall\ (k, j) \in A,\ i \in V \tag{6.11}$$
$$f_{ii} \geq 1 \qquad \forall\ i \in V \tag{6.12}$$

for the relaxation $P_p^{OC}(D)$ of the p-median polytope $p\,\mathrm{MP}(D)$.

Theorem 6.11. *$Q_p^{OC}(D)$ is an extended formulation of $P_p^{OC}(D)$.*

Proof. Let $(\bar{x}, \bar{y}) \in P_p^{OC}(D)$. We show that there exist $\bar{f}$ and $\bar{g}$ so that $(\bar{x}, \bar{y}, \bar{f}, \bar{g}) \in Q_p^{OC}(D)$ holds. All inequalities (6.1)–(6.3), (6.6), and (6.7) occur in both polytopes and they are not violated by $(\bar{x}, \bar{y})$. Assign the weight $\bar{w}_{ij} = 1 - 2\bar{x}_{ij}$ to every arc $(i, j) \in A$. Furthermore, define $\bar{f}_{ij}$ for all $i, j \in V$ and $\bar{g}_{ij}$ for all $i, j \in V$ as the weights of shortest odd and even walks, respectively, from vertex i to vertex j in D (assign a large value to the corresponding

variable if no such walk exists). Notice that, because there are no cycles with negative weight, a shortest even walk for every $i \in V$ to itself can simply use zero arcs and therefore has weight 0. This ensures that equations (6.9) hold. It is easy to see that inequalities (6.10) and (6.11) are not violated either due to the construction of $\bar{f}$ and $\bar{g}$. These inequalities can be interpreted as follows: The weight of shortest odd or even walks from vertex i to vertex j cannot exceed the weight of a shortest walk of the opposite parity from vertex i to vertex k plus the weight of the arc (k, j).

Inequalities (6.8) are satisfied by $(\bar{x}, \bar{y})$. This is, as mentioned above, equivalent to every odd cycle C having weight $\bar{w}(A_C)$ at least 1. Observe that $\bar{f}_{ii}$ is the weight of a shortest odd closed *walk* starting in $i \in V$. If this walk is not a cycle, there always exists an odd cycle with weight less or equal to the weight of a shortest odd closed walk. However, this cycle does not necessarily include vertex i, but its weight has to be as well greater or equal to 1. Therefore $\bar{f}_{ii} \geq 1$ holds for every $i \in V$.

For the converse, let $(\bar{x}, \bar{y}, \bar{f}, \bar{g}) \in Q_p^{OC}(D)$. Inequalities (6.9), (6.10), and (6.11) ensure that $\bar{f}_{ij}$ and $\bar{g}_{ij}$ are bounded from above by the weights of shortest odd and even walks, respectively, from vertex i to vertex j. Thus, the value $\bar{f}_{ii}$ is lower or equal than the weight of a shortest odd cycle through vertex i. Since $\bar{f}_{ii} \geq 1$ for every $i \in V$ by inequalities (6.12), no odd cycle inequality is violated and hence $(\bar{x}, \bar{y}) \in P_p^{OC}(D)$. □

Remark 6.12. *The extended formulation $Q_p^{OC}(D)$ of $P_p^{OC}(D)$ requires $m + n + 2n^2 - n = m + 2n^2$ variables. It has just $2mn + n$ additional inequalities compared to the natural relaxation $P_p(D)$, which ensure that all odd cycle inequalities are fulfilled.*

With Theorem 6.8, we can state the following consequence.

Corollary 6.13. *The p-median problem can be solved in polynomial time with the linear formulation $Q_p^{OC}(D)$ for every digraph $D = (V, A)$ and integer p if D fulfills every of the following criteria:*

- *D is triangle-free,*
- *D is oriented,*
- *D does not contain any of the digraphs H_1 nor H_2 as a subgraph, and*
- *D does not contain a nondirected g-odd weak Y-cycle $\tilde{C}$ and an arc (i, j) with $i, j \notin \tilde{C}$.*

Chapter 7

Solving the maximum clique problem on sparse graphs

As a central topic of this thesis, we study the maximum stable set problem, in particular for sparse instances. In Chapter 2, several inequalities are introduced that are valid for the stable set polytope. Many of them, such as the odd cycle inequalities and the 1-wheel inequalities, are the stronger the sparser the underlying graph G is. For dense graphs, the clique inequalities for maximal cliques are stronger. However, optimizing over the clique polytope ist $\mathcal{NP}$-hard, see Theorem 2.10.

Fortunately, we can solve the maximum stable set problem on dense graphs via fast, direct methods. Every stable set in a given graph G is a clique in its complementary graph $\bar{G}$. Therefore, we can find a maximum stable set in G by finding a maximum clique in $\bar{G}$. Since a dense graph implies a sparse complementary graph, we focus on maximum clique algorithms that are fast when applied to sparse graphs in this chapter.

The maximum clique problem has been studied extensively in the past and is still a very active research topic today. Many research results have led to a lot of publications dealing with various aspects of this problem. There exist plenty of fast algorithms, especially for sparse graphs and real-world graphs. Our contribution is to improve the algorithm of Pattabiraman et al. (2013). We illustrate the effect of the modifications we apply in a computational study.

7.1 Maximum clique algorithms: a literature review

A very simple idea that comes into mind for solving the maximum clique problem in a given graph $G = (V, E)$ is complete enumeration. Without having any information about the edge set E, every set in the power set of V except the empty set can potentially be a maximum clique in G. For every such candidate set $U \subseteq V$ of the vertices in G, one could check if $uv \in E$ for every pair $u, v \in U$.

The clique decision problem is one of the 21 original $\mathcal{NP}$-complete problems of Karp

(1972): In general, we are not able to check in polynomial time whether there exists a clique of size at least k in a given graph G unless $\mathcal{P} = \mathcal{NP}$. This implies that the maximum clique problem is $\mathcal{NP}$-hard.

Most algorithms for the maximum clique problem employ a branch-and-bound approach. Branching is used to systematically partition the search space for solution candidates. Fruitless search space can be discarded by *pruning* based on previously computed bounds. These bounds are very important for the performance of the algorithm. Usually, the size of the largest clique found so far is used as a lower bound for the clique number $\omega(G)$, whereas various techniques to compute upper bounds (globally for $\omega(G)$ and as well for possible clique sizes at each branching node in the branching tree) can be applied.

One of the first promising branch-and-bound algorithms for the maximum stable set problem was introduced by Carraghan and Pardalos (1990). It starts with an ordering $v_1, v_2, \ldots, v_n$ of the vertices in G such that $\deg(v_i) \leq \deg(v_j)$ for every $1 \leq i < j \leq n$. Then a depth first search is applied starting from vertex v_1 to find a largest clique in G that contains v_1. Thereafter, a depth first search starting from vertex v_2 is applied to the induced subgraph of G by $V \setminus \{v_1\}$. This procedure continues for all vertices v_k and the respective subgraph of G that is induced by $V \setminus \{v_1, \ldots, v_{k-1}\}$. The pruning strategy can be expressed in the following words: Check at each depth how many vertices are left as candidates to enlarge the current clique and backtrack whenever this number plus the depth (which is equal to the size of the current clique) is not greater than the size of the largest clique found so far. Carraghan and Pardalos (1990) observe that their initial ordering of the vertices by increasing degree is beneficial on dense graphs and that their algorithm is faster on sparse graphs when no vertex ordering is performed. A general advantage of this algorithm is that it allows for parallelization, which is useful to reduce the computational time.

A very good algorithm for random graphs and for DIMACS benchmark graphs from the *2nd DIMACS Implementation Challenge*, see Johnson and Trick (1996), is the algorithm of Östergård (2002). In contrast to the algorithm of Carraghan and Pardalos (1990), the ordering of vertices is reversed. First, cliques with vertex v_n in the subgraph of G induced by $V \setminus \{v_1, \ldots, v_{n-1}\}$ are considered, which is obviously the unique 1-clique $\{v_n\}$. This procedure continues for vertex v_{n-1} and the subgraph of G induced by $V \setminus \{v_1, \ldots, v_{n-2}\}$ and so on. Finally, cliques in G including vertex v_1 are considered. One idea behind this reversal is the following. The size of the best clique found so far can just be improved by 1 during the search at each step $i \in \{1, \ldots, n\}$ when considering cliques that include vertex $v_i \in V$. If such an improvement is made, one can immediately continue with vertex v_{i-1}. A modified pruning strategy in addition to the reversal makes this algorithm faster than the algorithm of Carraghan and Pardalos (1990). In order to further improve the performance, Östergård (2002) initially uses a degree-based greedy coloring and then orders the vertices according to their color classes. Beyond the efficiency of this algorithm to find a maximum clique in G, it allows to find *all* maximum cliques with a few small modifications.

Vertex colorings can be very useful to get bounds for the size of maximum cliques in a given graph. Notice that all vertices must be colored differently in any clique. One of the first algorithms which is essentially based on colorings, is presented by Babel (1991). It has turned out to be fast on arbitrary graphs. Besides computing bounds by using colorings,

branching rules are defined that use the information of these colorings. Babel (1991) shows that it has polynomial running time for a few graph classes such as linear running time for planar graphs. Later, Tomita and Seki (2003) present an algorithm that uses colorings and vertex ordering techniques to compute upper bounds. An improved version of their algorithm is given by Konc and Janežič (2007). Using dynamically varying bounds, it is significantly faster than the previous version.

Another useful characteristic number in graph theory for computing bounds in maximum clique algorithms is defined by Seidman (1983):

Definition 7.1. *The* k-core *of a graph* G*, if one exists, is the largest subgraph* H *of* G *such that* $\delta(H) \geq k$*.*

The k-core of G, for example, can be directly used to find upper bounds on the clique number $\omega(G)$. If G has no k-core, it cannot have a clique with more than k vertices. One can define the *core number* of a vertex with the definition from above, which is sometimes also called the *coreness* of a vertex, see Alvarez-Hamelin et al. (2005).

Definition 7.2. *The* core number *of a vertex* $v \in V$*, denoted by* $\kappa(v)$*, is the largest integer number such that* v *belongs to the* $\kappa(v)$*-core of* G*. The largest core number* $\kappa(v)$ *among all* $v \in V$ *is denoted by* $\kappa(G)$*.*

Abello et al. (1999) make use of bounds relying on k-cores in their algorithm. They are able to solve the maximum clique problem on graphs with millions of vertices, such as on a graph with more than 53 million vertices and more than 170 million (partially multiple) edges arising from telecommunications traffic data. A more recent, parallel maximum clique algorithm is proposed by Rossi et al. (2014). It starts with finding a large clique heuristically, which can enable for prunings at low depths in the branching tree. Thereafter, colorings and cores are used for aggressive pruning. Another algorithm that is based on k-cores is presented by Verma et al. (2015). After computing bounds, they employ the method of Östergård (2002). Two further interesting methods that rely on core-based bounds are presented by San Segundo et al. (2016, 2017), respectively. Both of them are designed for extremely large and sparse graphs and employ sparse encoding of adjacencies in G for speed-ups. The latter of the two algorithms is a parallelized version of the first one.

In this chapter, we focus on the algorithm of Pattabiraman et al. (2013), as it is really simple to implement and its strength does not depend on reordering vertices or computing potentially expensive bounds during the process. Some examples for frameworks where it is used are the following. Meghanathan (2015) determines the size of a maximal clique containing vertex v for every vertex $v \in V$ with an adjusted version of the algorithm. He then analyzes the distribution of maximal clique sizes in real-world and in random networks. Moreover, a similar approach is used by Meghanathan (2016) for an extensive analysis of the correlation between the size of a maximal clique and the so-called centrality metrics. Besides that, Gollub et al. (2017) make use of the original version of the maximum clique method of Pattabiraman et al. for efficient and reliable place recognition, which is an important task for enabling fully autonomous driving. This emphasizes the relevance of fast and simple algorithms for detecting a maximum or all maximal cliques.

A very interesting analysis of maximum clique algorithms, that were published prior to the publication of his work, is presented by Prosser (2012). He shows that the performance strongly depends on implementation techniques and that details can have enormous impact on the running time. Therefore, a comparison where all of the considered algorithms are implemented in the *java* programming language is drawn.

Although the maximum clique problem is a challenging problem in general and there are still unsolved benchmark instances on graphs with only 1,000 vertices, it can often be solved very fast in practice. Walteros and Buchanan (2018) address this discrepancy and observe that the running time strongly correlates with the difference between the clique number and the upper bound generated by the largest k such that G has a k-core. They call it the *clique-core gap* and show that it is often 0, 1, or 2 on real-life instances, whereas the clique core-gap of unsolved synthetic benchmark instances is on a three-digit level or even higher. They additionally point out that instances with small clique-core gaps are always easy to solve, whereas hard instances must have large clique-core gaps. Apart from this, their algorithm, which they apply for their computational experiments, is competitive with the best state-of-the-art algorithms in literature for instances with a clique-core gap of less than 100. Furthermore, it does not necessarily have a bad performance if the clique-core gap is 100 or more.

7.2 The algorithm of Pattabiraman et al.

As mentioned in the literature review, a lot of fast algorithms for the maximum clique problem were introduced since 1990. Pattabiraman et al. (2013) propose an algorithm that is very fast on sparse graphs and has a simple structure. They additionally present a heuristic, which runs many orders of magnitude faster than their algorithm and performs very well on their test set of large and sparse graphs. It can obviously not provide acceptable or even optimal solutions for arbitrary graphs in general. As we aim at optimal solutions, the heuristic is not dealt with here.

We assume that we are given an adjacency list for every vertex of G, where the vertices are listed in increasing order due to their index. Therefore, we use the convention that for every $i \in \{1, \dots, n\}$, the set $N(v_i)$ of all neighbors of vertex v_i is stored in a list which maintains the same order. Otherwise, we would do preprocessing to achieve this order, which would not be critical for the running time of the algorithm. Furthermore, we assume that we select the vertices in increasing order by their index when iterating over $N(v_i)$ for $i \in \{1, \dots, n\}$ in any for-loop. This is usually done anyway by the implementation, for example if $N(v_i)$ is stored in a list accessed from first to last. These few assumptions are necessary for some of the modifications we make in the upcoming section.

Algorithm 1 is the original algorithm of Pattabiraman et al. with a small adjustment. In line 9 of the subroutine CLIQUE$(G, U, size)$ we do not choose *any* vertex from U but the vertex with the smallest index instead. This does not produce additional computational costs because the vertex we choose is the vertex that was added first among the remaining vertices in U. Notice that the order in which vertices are added to U in line 9 of the main

routine MAXCLIQUE(G, lb) is determined by the order in which we access the vertices of $N(v_i)$ in line 6 and hence preserves the increasing order.

Algorithm 1
The maximum clique algorithm of Pattabiraman et al. (2013) with a fixed rule for choosing u in line 9 of the subroutine CLIQUE$(G, U, size)$.
Input: $G = (V, E)$, lower bound lb
Output: Size of a maximum clique

```
 1: procedure MAXCLIQUE(G, lb)
 2:     max ← lb
 3:     for i = 1 to n do
 4:         if deg(v_i) ≥ max then                          ▷ Pruning 1
 5:             U ← ∅
 6:             for each v_j ∈ N(v_i) do
 7:                 if j > i then                           ▷ Pruning 2
 8:                     if deg(v_j) ≥ max then              ▷ Pruning 3
 9:                         U ← U ∪ {v_j}
10:             CLIQUE(G, U, 1)

 1: procedure CLIQUE(G, U, size)
 2:     if U = ∅ then
 3:         if size > max then
 4:             max ← size
 5:         return
 6:     while |U| > 0 do
 7:         if size + |U| ≤ max then                        ▷ Pruning 4
 8:             return
 9:         v_j ← argmin{i : v_i ∈ U}
10:         U ← U\{v_j}
11:         N'(v_j) := {v_l : v_l ∈ N(v_j) and deg(v_l) ≥ max}   ▷ Pruning 5
12:         CLIQUE(G, U ∩ N'(v_j), size + 1)
```

As an input parameter, Algorithm 1 requires a lower bound lb for the size of a maximum clique in G, whose default value is 0. We are able to set lb to 2 if $E \neq \emptyset$ or to any other number if we have any certificate that a clique of the respective value exists. Throughout, the variable max stores the size of the best clique found so far and the variable $size$ states the depth of the search which is the size of the clique that is considered currently. When the algorithm terminates, the value max equals the clique number $\omega(G)$.

There are five pruning opportunities in the algorithm. Three of them appear in the main routine and the other two in the subroutine. The procedure MAXCLIQUE(G, lb) consecutively searches for cliques that contain a fixed vertex v_i for $i \in \{1, \ldots, n\}$ and vertices whose index is greater than i. `Pruning 1` immediately discards vertex v_i if the

number of its neighbors is lower than the size of the largest clique found so far. If vertex v_i cannot be ruled out at this step, a candidate list $U \subseteq N(v_i)$ is built. `Pruning 2` excludes vertices with a lower index, since this avoids computing cliques that have been considered previously. When searching for the size of the largest clique including vertex v_i, we already have considered all candidates that include vertex v_j for all $j \in \{1, \ldots, i-1\}$. The second requirement for a candidate vertex to be added to U is that it has *enough* neighbors. Thus, its degree also has to be at least equal to max, see `Pruning 3`. The remaining prunings appear in the procedure CLIQUE$(G, U, size)$. `Pruning 4` causes backtracking if the current clique's size plus the number of candidates in U to enlarge this clique does not exceed the size of the best clique found so far. If we do not track back at this point, we choose v_j, the vertex with the lowest index from U, to extend the current clique. The set $N'(v_j)$ then is given by the neighbors of v_j that have a sufficient high degree, see `Pruning 5`. Intersected with U, it yields the set of all candidates to further enlarge the current clique. The algorithm continues with calling CLIQUE$(G, U \cap N'(v_j), size + 1)$. Again, we can assume that $N'(v_j)$ and $U \cap N'(v_j)$ maintain the increasing order of elements with respect to their indices.

We recognize that every pruning uses information which is already available at the time it is called: max and $size$ are variables; index, neighborhood, and degree of all vertices are known right from the start of the algorithm; the cardinality of U is either known or easy to determine.

7.3 Modifications and computational experiments

Our aim is to reduce the running time of Algorithm 1 while maintaining its simplicity. The vertex degree is used in `Pruning 1`, `3`, and `5` to discard fruitless search space. We suggest to use alternative parameters for each vertex to replace the vertex degree in the appropriate pruning steps. These parameters should be very quick to compute once before the algorithm starts and the replacement of the vertex-degree-based prunings should be possible with not more than a few adjustments.

7.3.1 The upper degree approach

The main idea of Algorithm 1 is that the current clique, possibly the empty set, can only be enlarged with vertices that have a sufficient high degree. Let us say we are starting with vertex v_i and try to improve the largest clique found so far. As we consider vertices in increasing order regarding their indices, we previously have considered all candidates for a maximum clique that contain one (or more) of the vertices $v_1, \ldots, v_{i-1}$. During the ongoing search, the degree of some of the remaining candidates $v_i, \ldots, v_n$ might be sufficiently large to not be pruned, but adjacencies to vertices from the set $\{v_1, \ldots, v_{i-1}\}$ are actually irrelevant. Hence, we do not only require every vertex $v_j \in \{v_i, \ldots, v_n\}$ to have a sufficient number of neighbors in V, but it must have enough neighbors in the set $\{v_i, \ldots, v_n\}$.

Definition 7.3. *The* upper degree *of vertex $v_i \in V$ with respect to the initial ordering,*

denoted by $\deg^{\uparrow}(v_i)$*, is given by the number of neighbors of* v_i *whose index is greater than* i*:*

$$\deg^{\uparrow}(v_i) = |N(v_i) \cap \{v_{i+1}, \ldots, v_n\}|.$$

The upper neighborhood *of vertex* v_i *is the set of vertices in* $N(v_i) \cap \{v_{i+1}, \ldots, v_n\}$.

Algorithm 2
A modification of Algorithm 1 that uses the upper degree of vertices for prunings whenever possible.
Input: $G = (V, E)$, lower bound lb
Output: Size of a maximum clique

```
1:  procedure MAXCLIQUE(G, lb)
2:      max ← lb
3:      for i = 1 to n do
4:          if deg↑(v_i) ≥ max then                          ▷ Pruning 1
5:              U ← ∅
6:              k ← 1
7:              for each v_j ∈ N(v_i) do
8:                  if j > i then                            ▷ Pruning 2
9:                      if k + deg↑(v_j) ≥ max then          ▷ Pruning 3
10:                         U ← U ∪ {v_j}
11:                         k ← k + 1
12:             CLIQUE(G, U, 1)

1:  procedure CLIQUE(G, U, size)
2:      if U = ∅ then
3:          if size > max then
4:              max ← size
5:          return
6:      while |U| > 0 do
7:          if size + |U| ≤ max then                         ▷ Pruning 4
8:              return
9:          v_j ← argmin{i : v_i ∈ U}
10:         U ← U\{v_j}
11:         N' ← ∅
12:         k ← 1
13:         for each v_l ∈ N(v_j) do
14:             if l > j then                                ▷ Pruning 5a
15:                 if size + k + deg↑(v_l) ≥ max then       ▷ Pruning 5b
16:                     N' ← N' ∪ {v_l}
17:                     k ← k + 1
18:         CLIQUE(G, U ∩ N', size + 1)
```

In Algorithm 2, every pruning based on the vertex degree in Algorithm 1 is replaced by a stronger condition that uses the upper degree. `Pruning 1` allows replacing $\deg(v_i)$ by $\deg^{\uparrow}(v_i)$ without any adjustment to be made. Every vertex whose index is greater than i and which is in the neighborhood of v_i is a candidate for an improvement. On the contrary, `Pruning 3` needs to be adjusted. When the elements in $N(v_i)$ whose index is greater than i, see line 7 and 8 of Algorithm 2, are considered, we assume that this happens in increasing order. The first vertex we pick in line 7 and that survives `Pruning 2`, say v_j, requires a sufficient number of neighbors with a higher index to be a candidate for improvement. Because the current clique we consider already includes vertex v_i, a clique that improves the best clique found so far, which has cardinality max, must consist of at least $max - 1$ more vertices together with v_j. If this is the case, we put it into the candidate set U. Notice that the presence of vertex v_i is the reason for initializing variable k with value 1. While running the for-loop in line 7, every vertex must have as many neighbors with a higher index such that it could improve the best clique found so far together with v_i, itself, and all candidates that are currently present in U. Variable k increases by 1 whenever a vertex is added to U and hence ensures that `Pruning 3` works correctly. In order to similarly adjust `Pruning 5`, it is divided into `Pruning 5a` and `5b`. In Algorithm 1, we add all vertices v_l to $N'(v_j)$ which are in the neighborhood of v_j and concurrently have a sufficient high degree. Notice that we recently added vertex v_j to the current clique which previously had the size stored in the variable $size$. Again, due to the order in which we consider vertices in the neighborhood set, we only require vertices with a higher index as candidates for improvement, see `Pruning 5a`. We denote this candidate set by N' and a vertex v_l is added to N' if and only if the union of the current clique (which already includes vertex v_j), all vertices in N', and the vertices in the upper neighborhood of v_l has more vertices than the best clique found so far. Variable k encodes the number of vertices that are currently present in $N' \cup \{v_j\}$, as the variable $size$ has not been updated yet while v_j has been added to the current clique.

All in all, we do not take adjacencies to discarded vertices into account anymore with our modified prunings in Algorithm 2, but we have an additional cost in contrast to Algorithm 1, which is caused by computing the upper degree for every vertex in the graph. Nevertheless, applying this preprocessing is quick and worthwhile. We will verify this with our numerical results in Section 7.3.3.

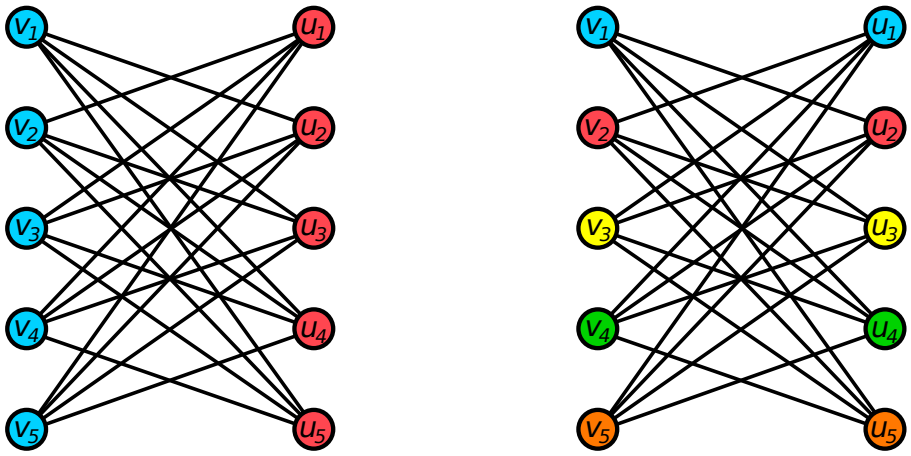
7.3.2 Applying greedy colorings

We have strengthened three of the five pruning steps from Algorithm 1 by replacing the vertex degree conditions by upper vertex degree conditions in the previous section. Next, we further improve the strengths of `Pruning 1`, `3`, and `5` by choosing another number which is determined for each vertex v in G. Among several possibilities, the number of different colors that are assigned to the neighbors of v by a coloring in G seems to be promising when trying to find strong bounds that can be determined in reasonable time. Thus, we restrict ourselves to coloring-based bounds in what follows. Finding the chromatic number in a given graph G is an $\mathcal{NP}$-hard problem and we are not able to find an optimal coloring in G in reasonable time in general. Therefore, we are interested in an approximation of an

optimal coloring which has an acceptable approximation quality.

Matula et al. (1972) define a sequential coloring as a coloring of vertices $\{v_1, \ldots, v_n\}$ in G, where color 1 is assigned to v_1 and each succeeding vertex is colored with the smallest color number such that a feasible coloring is maintained when assigning a color to v_n. Since this is a greedy strategy, we simply call it a *greedy coloring*.

The approximation quality of a greedy coloring depends on the order in which the vertices are colored. For every graph G, there exists an ordering of the vertices such that a greedy coloring gives an optimal coloring in G, cf. Husfeldt (2015). Mitchem (1976) compares algorithms that all apply greedy colorings, but differ in how the vertices are ordered previously. He shows that for each of these algorithms, there exist graphs such that they behave arbitrarily bad. Johnson (1974) illustrates that a graph providing a worst case example is the *crown graph*, see Figure 7.1(a) and 7.1(b).



(a) An optimal coloring.

(b) A worst case greedy coloring.

Figure 7.1: Two different greedy colorings depending on the vertex ordering.

For even n, we define the vertex set of the crown graph to be $V := \{v_1, \ldots, v_{n/2}\} \dot\cup \{u_1, \ldots, u_{n/2}\}$. It is bipartite and vertex v_i is adjacent to vertex u_j if and only if $i \neq j$. In our example, we have $n = 10$. An optimal coloring uses two colors and is achieved by the ordering $v_1, \ldots, v_5, u_1, \ldots, u_5$ for the greedy coloring. Otherwise, it is possible that a greedy coloring uses $\frac{n}{2} = 5$ colors. This is the case if the vertex ordering is for instance $v_1, u_1, \ldots, v_5, u_5$.

The number of colors that are used by a greedy coloring is an upper bound for the clique number $\omega(G)$, since every vertex in a clique needs to be colored with a different color. Our aim is to replace the vertex degree conditions by information we get from greedy colorings, so we are searching for an upper bound on the size of cliques that contain a specific vertex. Every vertex $v \in V$ has a different color than its neighbors. If some of the neighbors of v

share the same color, they are definitely not connected by an edge and hence at most one of these vertices can be concurrently present in a clique together with vertex v.

Lemma 7.4. *Let G be colored with an arbitrary coloring. The size of any clique containing a specific vertex v of G cannot exceed the number of different colors in $N(v)$ by more than 1.*

Proof. Let v be a vertex in G. All vertices in a clique have to be colored with a different color, since vertices with the same color are not connected by an edge. Therefore, the number of different colors in $N(v)$ is an upper bound for the size of cliques in $N(v)$. Vertex v itself has a different color than all of its neighbors and enlarges cliques in $N(v)$ by 1. □

Every vertex in Figure 7.1(a) has just one color in its neighborhood, so 2 is an upper bound on the maximal clique size for each vertex. This upper bound is 5 for every vertex in Figure 7.1(b). Although this is the worst bound possible with respect to coloring bounds, it is not worse than the degree bound. We certainly hope to get much better bounds in practice.

Let us consider another small graph, which has no uniform structure like the crown graph. The priority of the colors we use is 1 =*blue*, 2 =*red*, 3 =*yellow*, 4 =*green*, and 5 =*orange* as in the crown graph example. We maintain this order for all upcoming examples.

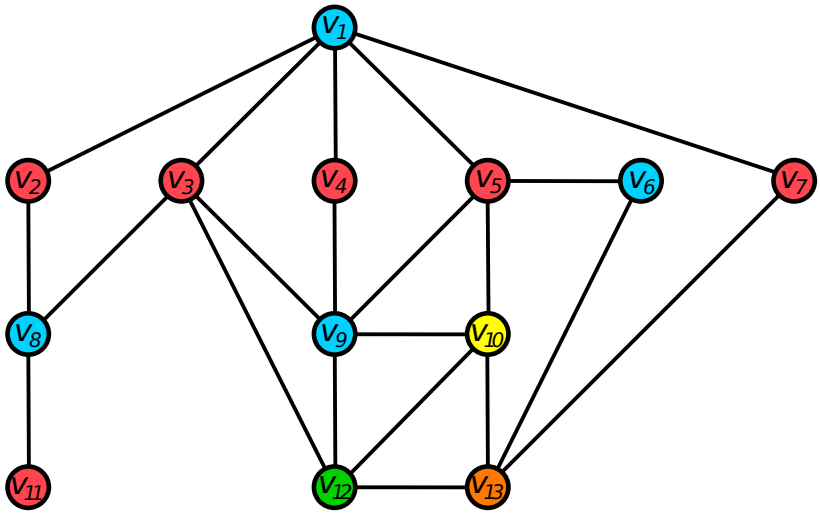


Figure 7.2: A greedy coloring. Vertices are sorted in increasing order regarding their index: $v_1, \ldots, v_{13}$.

In the graph in Figure 7.2, the order in which the vertices are colored via a greedy coloring is $v_1, \ldots, v_{13}$. We obtain that we use five different colors in total, so that we can bound $\omega(G)$ from above by 5. Taking a closer look at some vertices, we see that we get a tight

upper bound of 2 for vertex v_1 and a bad upper bound of 5 for vertex v_{12}. As a vertex itself has a different color than its neighbors, the upper bound for this vertex regarding the clique size is the number of different colors we find in its neighborhood plus 1.

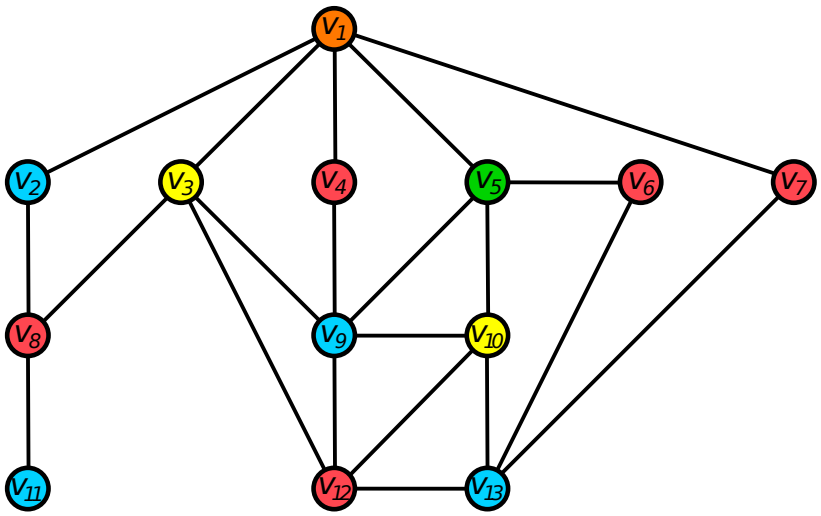


Figure 7.3: A greedy coloring: Vertices are sorted in decreasing order regarding their index: $v_{13}, \ldots, v_1$.

Figure 7.3 shows the same graph as Figure 7.2, but the vertices are colored in reverse order, i.e. $v_{13}, \ldots, v_1$. By chance, we get the same upper bound for $\omega(G)$, but different bounds for the maximal clique sizes that contain specific vertices. The upper bound for the maximal clique size containing vertex v_1 is now 5, whereas vertex v_{12} just has the colors blue and yellow in its neighborhood, so the upper bound for v_{12} is 3.

We have applied two greedy colorings to the same graph with two different vertex orderings. The vertices are sorted in increasing and decreasing order, respectively, concerning their indices. We have seen that for vertices v_1 and v_{12}, we get varying bounds. The idea is now to choose the best bound for each vertex.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
$ub1$	2	2	3	2	3	3	3	2	4	5	2	5	5
$ub2$	5	3	4	3	5	3	3	3	4	4	2	3	3
$\min\{ub1, ub2\}$	2	2	3	2	3	3	3	2	4	4	2	3	3

Table 7.1: Bounds that arise from the greedy colorings in Figures 7.2 and 7.3.

As we observe in Table 7.1, we are able to improve every upper bound of 5 in this example.

Additionally, we often have a difference of 1 or 2 in the columns when comparing values of $ub1$ and $ub2$, which are the upper bounds that arise from the forward coloring visualized in Figure 7.2 and the backward coloring visualized in Figure 7.3, respectively. Upper bound $ub1$ delivers good bounds for lower indexed vertices and bad bounds for higher indexed vertices. Moreover, the reverse is true for $ub2$. The reason should be that at the end of each coloring we have less degrees of freedom and are forced to use higher order colors for the vertices we color last.

Next, we want to compare the bounds we get by the two primitive greedy colorings to the bounds that an optimal coloring provides.

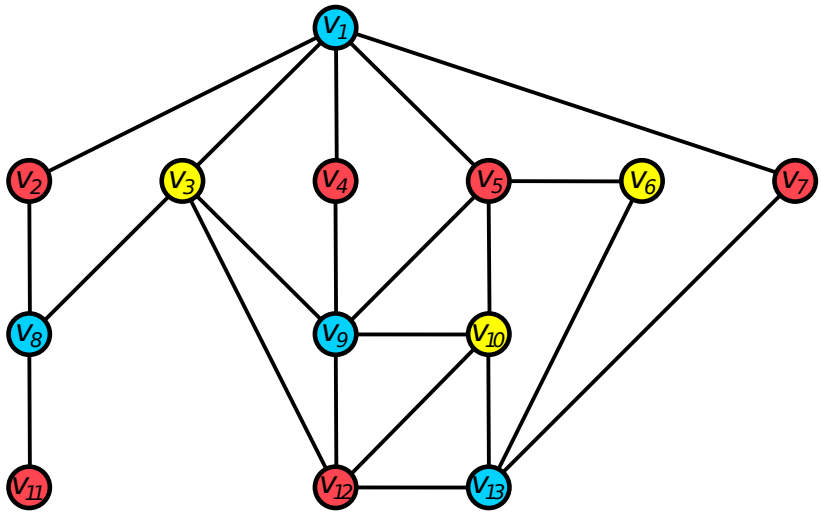


Figure 7.4: An optimal coloring. A greedy coloring with vertex order $v_1, v_{13}, v_{12}, v_2, \ldots, v_{11}$ yields the same coloring.

An optimal coloring is presented in Figure 7.4. Notice that it is achieved by a greedy coloring when the vertex order is $v_1, v_{13}, v_{12}, v_2, \ldots, v_{11}$. In contrast to the greedy colorings from above, we only use the colors *blue*, *red*, and *yellow*. Hence, the maximum clique size $\omega(G)$ is bounded by 3. We are interested in how strong the "cheap" vertex bounds obtained from the combination of $ub1$ and $ub2$ are, compared to the vertex bounds we get from an optimal coloring.

Table 7.2 shows that the combined information we get from $ub1$ and $ub2$ is competitive with the bounds $ub3$ of the optimal coloring in Figure 7.4. For some vertices the bounds are better, for others they are worse. However, none of these bounds differs by more than 1 for each vertex $v \in V$ here. Another interesting observation regarding the example is that the majority of the bounds is tight. Notice that no upper bound can be lower than the size of a largest clique including the respective vertex.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
degree	6	3	5	3	5	3	3	4	6	5	2	5	5
$\min\{ub1, ub2\}$	2	2	3	2	3	3	3	2	4	4	2	3	3
$ub3$	3	2	3	2	3	3	2	3	3	3	2	3	3
largest clique size	2	2	3	2	3	2	2	2	3	3	2	3	3

Table 7.2: Bounds that arise from the greedy colorings in Figures 7.2 and 7.3.

Algorithm 3
A modification of Algorithm 1 that uses the number of colors in the neighborhood of vertices for prunings whenever possible.
Input: $G = (V, E)$, lower bound lb
Output: Size of a maximum clique

```
1:  procedure MAXCLIQUE(G, lb)
2:      max ← lb
3:      for i = 1 to n do
4:          if ncn(v_i) ≥ max then                          ▷ Pruning 1
5:              U ← ∅
6:              for each v_j ∈ N(v_i) do
7:                  if j > i then                           ▷ Pruning 2
8:                      if ncn(v_j) ≥ max then              ▷ Pruning 3
9:                          U ← U ∪ {v_j}
10:             CLIQUE(G, U, 1)

1:  procedure CLIQUE(G, U, size)
2:      if U = ∅ then
3:          if size > max then
4:              max ← size
5:          return
6:      while |U| > 0 do
7:          if size + |U| ≤ max then                        ▷ Pruning 4
8:              return
9:          v_j ← argmin{i : v_i ∈ U}
10:         U ← U\{v_j}
11:         N'(v_j) := {v_l : v_l ∈ N(v_j) and ncn(v_l) ≥ max}   ▷ Pruning 5
12:         CLIQUE(G, U ∩ N'(v_j), size + 1)
```

In order to modify the algorithm of Pattabiraman et al. (2013), we generalize the idea

from the example above and introduce the following notation.

Definition 7.5. *Let the vertices of G be colored by one or more greedy colorings. The* number of colors in the neighborhood *of vertex $v_i \in V$, denoted by* $\mathrm{ncn}(v_i)$*, is the minimal number of different colors that are assigned to vertices in the neighborhood $N(v_i)$ of vertex v_i among all greedy colorings performed.*

If exactly one coloring is used, the number of colors in the neighborhood is also known as the *saturation degree* and is used for example in the maximum clique algorithm of Babel (1991). Therefore, we can consider our approach as a generalization of the saturation degree.

We substitute the degree by the number of colors in the neighborhood in `Pruning 1`, `3`, and `5` of Algorithm 1 to obtain Algorithm 3. Applying one or more greedy colorings as a preprocessing of Algorithm 3 is not for free and can be the bottleneck of its running time. We will go into detail in the upcoming section where we compare all the algorithms we suggest. The number of colors in the neighborhood of a vertex can be substantially lower than its degree and thus can deliver a much better bound for the size of a clique containing this vertex.

Amongst others, we randomly generated many large graphs with an average vertex degree of 4 and 20, respectively, for our experiments in Section 7.3.3. For these graphs, the average number of colors in the neighborhood of the vertices with an average degree of 4 is just 2 when applying one forward and one backward coloring. Moreover, the average number of colors in the neighborhood of the vertices in graphs with an average vertex degree of 20 is 7. Even in the graph in Figures 7.2–7.4, which has only 13 vertices, we have significant deviations, see Table 7.2.

We have deduced two types of changes that can be implemented to strengthen three of five prunings in Algorithm 1. The first one is to still consider some kind of degree of each vertex while only taking neighbors with a higher index into account. This is possible because of the ordering in which clique candidates are enumerated. Thus, Algorithm 2 works correctly. The second type of changes, which we made to formulate Algorithm 3, is to use stronger bounds for each vertex than the degree bounds. An upper bound for some vertex relies on the number of different colors assigned to all of its neighbors. Clearly, both modifications can be combined.

For each vertex in the graph in Figures 7.2–7.4, we compare the number of different colors in its upper neighborhood with its upper degree, that we use for Algorithm 2, and only consider cliques where the current vertex has the lowest index.

The bounds $ub1'$, $ub2'$, and $ub3'$, respectively, in Table 7.3 arise from the forward, the backward, and the optimal coloring in Figures 7.2, 7.3, and 7.4. Each of these bounds is defined as the number of different colors that are used for the vertex itself and the vertices in its upper neighborhood. The size of a largest clique containing vertex v_i, $i \in \{1, \ldots, 13\}$, and only vertices in its upper neighborhood is given in the last row of Table 7.3. Observe that the upper degree bound is tight for vertices $v_2, v_4, v_6, \ldots, v_{13}$, but not for the remaining vertices. The bound $ub3'$ from the optimal coloring is tight except for v_1. Choosing the best bound from $ub1'$ and $ub2'$ yields a tight bound for *every* vertex. Obviously, the quality

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
upper degree	6	2	4	2	4	2	2	2	3	3	1	2	1
$ub1'$	2	2	3	2	3	2	2	2	3	3	1	2	1
$ub2'$	5	2	3	2	4	2	2	2	3	3	1	2	1
$\min\{ub1', ub2'\}$	2	2	3	2	3	2	2	2	3	3	1	2	1
$ub3'$	3	2	3	2	3	2	2	2	3	3	1	2	1
largest clique size with upper neighbors	2	2	3	2	3	2	2	2	3	3	1	2	1

Table 7.3: Bounds for the size of a largest clique where v_i is the vertex with the lowest index, arising from the greedy colorings in Figures 7.2, 7.3, and 7.4.

depends on the graph structure and we only use a tiny graph for illustration here. However, it prompts us to apply this strategy when solving large instances.

Definition 7.6. *Let the vertices of G be colored by one or more greedy colorings. The* number of colors in the upper neighborhood *of vertex $v_i \in V$, denoted by* $\mathrm{ncn}^{\uparrow}(v_i)$*, is the minimal number of different colors that are assigned to vertices in the upper neighborhood $N(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ of vertex v_i among all greedy colorings performed.*

As a preprocessing of Algorithm 4 we have to perform one or more greedy colorings. For each vertex v_i, we just have to count how many different colors are assigned to vertices in the set $\{v_{i+1}, \ldots, v_n\}$.

The information type of the upper degree and the number of colors in the upper neighborhood is the same: It provides us a bound on how many neighbors of a vertex with a higher index can be concurrently present in a clique together with the vertex itself. Thus, we can simply replace every $\deg^{\uparrow}$ by $\mathrm{ncn}^{\uparrow}$ in Algorithm 2 to obtain Algorithm 4.

Algorithm 4
A modification of Algorithm 1 that uses the number of colors in the upper neighborhood of vertices for prunings whenever possible.
Input: $G = (V, E)$, lower bound lb
Output: Size of a maximum clique

```
1:  procedure MAXCLIQUE(G, lb)
2:      max ← lb
3:      for i = 1 to n do
4:          if ncn↑(v_i) ≥ max then                          ▷ Pruning 1
5:              U ← ∅
6:              k ← 1
7:              for each v_j ∈ N(v_i) do
8:                  if j > i then                            ▷ Pruning 2
9:                      if k + ncn↑(v_j) ≥ max then          ▷ Pruning 3
10:                         U ← U ∪ {v_j}
11:                         k ← k + 1
12:             CLIQUE(G, U, 1)

1:  procedure CLIQUE(G, U, size)
2:      if U = ∅ then
3:          if size > max then
4:              max ← size
5:          return
6:      while |U| > 0 do
7:          if size + |U| ≤ max then                         ▷ Pruning 4
8:              return
9:          v_j ← argmin{i : v_i ∈ U}
10:         U ← U\{v_j}
11:         N' ← ∅
12:         k ← 1
13:         for each v_l ∈ N(v_j) do
14:             if l > j then                                ▷ Pruning 5a
15:                 if size + k + ncn↑(v_l) ≥ max then       ▷ Pruning 5b
16:                     N' ← N' ∪ {v_l}
17:                     k ← k + 1
18:         CLIQUE(G, U ∩ N', size + 1)
```

7.3.3 Computational results

We already mentioned that the running time of a maximum clique algorithm highly depends on the implementation technique. It should be pointed out that the source code of most of the existing maximum clique algorithms in literature is insofar optimized as it uses fast programming languages and appropriate techniques for accessing and saving data. The goal of our computational study is to compare Algorithm 1 of Pattabiraman et al. (2013) with our three modified Algorithms 2, 3, and 4 under fair conditions. Therefore, we have implemented Algorithms 1–4 in Python and use the package NetworkX, see Hagberg et al. (2008), for accessing and saving information of the graph.

When reading a graph from a file, we initially transform the information into an ordered adjacency list. This process has almost no influence on the total running time. Of course, the preprocessing, which has to be made to compute the upper degree for Algorithm 2, the number of colors in the neighborhood for Algorithm 3, and the number of colors in the upper neighborhood for Algorithm 4, might possibly be a crucial factor for the total running time. Notice that we apply one forward and one backward coloring with respect to the vertex order, when determining the number of colors in the (upper) neighborhood.

To have a representative set of large and sparse test instances, we randomly generated test instances with the graph generator *gnm_random_graph* from NetworkX with fixed parameters n and m. The parameter n takes several values between 1,000 and 50,000. Sparsity is controlled by choosing m such that the expected average vertex degree is at most 20. A consequence of this choice is that $\omega(G) = 3$ for most of the graphs and $\omega(G) = 4$ for the others. Our study additionally includes nine test instances from the 2nd DIMACS Implementation Challenge, which are relatively sparse.

Besides comparing the running time, we count the number of vertices that can be pruned by `Pruning 1` for each of the algorithms we consider. This should give us an impression of how strong the respective pruning is. It is not very meaningful to count the vertices pruned by `Pruning 3` and `5` because they are not independent of `Pruning 1`. Whenever we prune at an early stage, a lot of candidates for later prunings are eliminated.

For each pair of parameters n and m, ten test instances are randomly generated. Table 7.4 includes the average CPU times in seconds and the average numbers of nodes in the search tree that are pruned during the enumeration process by `Pruning 1`. Moreover, the average CPU times for Algorithms 3 and 4, respectively, without the running times consumed by the preprocessing are presented in columns Algo3* and Algo4*. The reason to display them separately is that Algorithms 3 and 4 get relatively slow on larger instances. Whenever an algorithm is the fastest among Algorithms 1–4, its running time in the respective column is given in bold numbers.

We can see that Algorithm 2 is the fastest one in every test set of ten instances. It is many times faster than Algorithm 1 for every pair of parameters n and m. Unfortunately, Algorithms 3 and 4 are not competitive against the others, although the prunings are much stronger, which is indicated by the number nodes in the search tree pruned by `Pruning 1` in the last four columns of the table. Their total running times are many times higher than those of Algorithms 1 and 2, which is caused by computing the coloring-based bounds.

		CPU (s)						Pruning 1 nodes			
n	m	Algo1	Algo2	Algo3	Algo4	Algo3*	Algo4*	Algo1	Algo2	Algo3	Algo4
1,000	5,000	0.10	**0.03**	0.32	0.36	0.04	0.04	3	298	31	482
1,000	10,000	0.33	**0.06**	0.67	0.72	0.13	0.13	0	194	0	343
2,000	5,000	0.05	**0.02**	0.65	0.72	0.06	0.07	247	1,128	1,122	1,716
2,000	10,000	0.17	**0.03**	1.16	1.26	0.04	0.04	5	603	67	980
2,000	20,000	0.67	**0.11**	2.21	2.41	0.12	0.10	0	383	0	661
5,000	10,000	0.08	**0.03**	3.09	3.47	0.06	0.07	1,158	3,287	3,800	4,600
5,000	20,000	0.28	**0.07**	5.72	6.42	0.10	0.10	69	1,872	635	3,054
5,000	50,000	1.75	**0.22**	13.45	14.77	0.29	0.23	0	784	0	1,243
10,000	20,000	0.17	**0.07**	12.45	14.01	0.12	0.14	2,357	6,600	7,668	9,280
10,000	50,000	0.91	**0.16**	28.37	31.60	0.16	0.15	28	3,007	368	4,914
10,000	100,000	3.81	**0.47**	54.48	60.40	0.58	0.44	0	1,499	0	2,353
20,000	50,000	0.50	**0.15**	60.95	67.90	0.11	0.15	2,449	11,275	11,261	17,142
20,000	100,000	1.95	**0.33**	114.95	128.61	0.37	0.34	57	5,983	716	9,809
20,000	200,000	8.14	**1.03**	221.92	249.03	1.32	1.00	0	3,007	1	4,725
50,000	100,000	0.99	**0.43**	332.87	372.26	0.23	0.29	11,709	32,976	38,378	46,406
50,000	200,000	3.35	**0.74**	677.56	753.80	0.59	0.56	665	18,661	6,423	30,659
50,000	500,000	22.14	**3.00**	1,669.73	1,902.06	3.52	2.64	0	7,486	1	11,764

Table 7.4: For randomly generated instances with fixed n and m, the average running times and the average numbers of nodes in the search tree pruned by `Pruning 1` over ten instances are presented.

Although there is potential for improvements by more efficient implementation techniques, it is very unlikely to keep up with the two algorithms that are based on degree prunings. Even if the preprocessing time would be 0 for Algorithms 3 and 4, they would not be significantly faster than Algorithm 2 when applied to our set of test instances.

The question arises whether Algorithms 3 and 4 are useless in practice. Algorithm 1 is fast on massive sparse graphs and, like our suggested modified algorithms, easy to implement. We constructed an improved version, denoted by Algorithm 2, via substituting degree prunings by upper degree prunings. If the underlying graph is more dense, Algorithms 1 and 2 get slower in general and coloring-based prunings may be much stronger than degree-based prunings. Moreover, the preprocessing times of Algorithms 3 and 4 highly depend on the number of vertices.

				CPU (s)				Pruning 1 nodes			
Instance	n	m	$\omega(G)$	Algo1	Algo2	Algo3	Algo4	Algo1	Algo2	Algo3	Algo4
c-fat200-1	200	1,534	12	0.01	$<$ **0.01**	0.03	0.03	0	148	199	199
c-fat200-2	200	3,235	24	0.06	**0.01**	0.05	0.05	0	144	199	199
c-fat200-5	200	8,473	58	13.69	2.1	3.58	**0.22**	0	134	0	173
c-fat500-1	500	4,459	14	0.04	$<$ **0.01**	0.14	0.17	0	399	499	499
c-fat500-2	500	9,139	26	0.33	**0.05**	0.28	0.34	0	359	499	499
c-fat500-5	500	23,191	64	63.26	9.77	**0.74**	0.82	0	337	499	499
p_hat300-1	300	10,933	8	10.47	**0.74**	1.45	0.95	0	28	0	56
p_hat500-1	500	31,569	9	178.88	**12.53**	24.00	12.94	0	43	0	75
p_hat700-1	700	60,999	11	957.16	69.99	127.70	**69.65**	0	56	0	99

Table 7.5: Instances from the 2nd DIMACS Implementation Challenge.

Table 7.5 includes DIMACS benchmark graphs with less than 1,000 vertices. These graphs are synthetically constructed and contain (very) large cliques. The instance with the smallest maximum clique among the instances from Table 7.5 is *p_hat300-1* with $\omega(G) = 8$. For *c-fat500-5*, which is the instance with the largest maximum clique, we have $\omega(G) = 64$. We observe again that Algorithm 2 is many times faster than Algorithm 1 on every instance. Interestingly, there are instances where Algorithms 3 and 4 are much faster. The largest clique numbers among the instances we consider are 58 and 64, respectively, for *c-fat200-5* and *c-fat500-5*. In both cases, Algorithm 4 is about ten times faster than Algorithm 2.

To draw a conclusion, our suggested Algorithm 2 works very well when applied to massive sparse graphs. Having graphs with less vertices and a higher density, computing bounds $\mathrm{ncn}^{\uparrow}(v)$ for every vertex $v \in V$ seems to be worthwhile. Then we prefer Algorithm 4, which is given by substituting every $\deg^{\uparrow}$ by $\mathrm{ncn}^{\uparrow}$ in Algorithm 2.

Chapter 8

Conclusion

In this thesis, we have presented new extended formulations for several optimization problems. Besides proving their correctness, computational experiments for some of them have been performed to illustrate their strength or to compare the running time for solving them to the running time of previously known formulations. Moreover, we were able to improve a fast branch-and-bound algorithm for the maximum clique problem on very large sparse graphs, which is simple to implement and hence predestined for practical use.

Some relaxations for the maximum stable set problem, such as the odd cycle polytope and the 1-wheel polytope, can contain exponentially many inequalities that can be separated in polynomial time. Then, a polynomial size extended formulation allows for optimizing over the original exponential size formulation. An extended formulation of the odd cycle polytope was given by Yannakakis (1991). We have introduced an alternative extended formulation which has less inequalities and is faster in practice. Every instance in our set of test instances was solved in no more than 212 seconds with our new formulation, whereas some instances have not been solved within our time limit of an hour with the old formulation. Moreover, we achieved average time savings of 24% over all test instances that were solved by both formulations within the time limit. For the 1-wheel polytope of the maximum stable set problem, that contains the odd and even 1-wheel inequalities, we have presented the first compact extended formulation. It is based on the separation algorithm of de Vries (2015). Furthermore, we have shown how to construct an extended formulation based on the separation algorithm of Cheng and Cunningham (1997), which is competitive on very dense graphs.

The A-odd cycle inequalities of the Boolean Quadric Polytope (BQP) can be used to obtain a very strong relaxation of the nonconvex quadratic program with box constraints (BoxQP). We have shown how to construct a tight compact extended relaxation for the BoxQP by strengthening a weak linear relaxation for the BoxQP with our extended formulation for the A-odd cycle inequalities. For illustration, we performed computational experiments on a large benchmark set.

The odd cycle relaxation for the p-median problem contains the directed odd cycle in-

equalities and it is known to be integral if the underlying graph has specific properties. In this case, one is able to solve the p-median problem in polynomial time. Therefore, we have constructed an extended formulation of the polytope that contains all directed odd cycle inequalities.

In the last part of this thesis, we have studied and improved an enumerative branch-and-bound algorithm for the maximum clique problem, which is fast for very large sparse graphs. We suggested three modifications and observed that one of them is many times faster than the original version. The two remaining versions are based on bounds that originate from greedy colorings instead of using degree based bounds. We prefer them if the underlying graph has less vertices and a higher density.

Bibliography

Abello, J., Pardalos, P. M., and Resende, M. G. C. (1999). On maximum clique problems in very large graphs. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 50: 119–130.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Applications and Algorithms*. Prentice-Hall, Upper Saddle River, New Jersey.

Alvarez-Hamelin, J. I., Dall'Asta, L., Barrat, A., and Vespignani, A. (2005). k-core decomposition: A tool for the visualization of large scale networks. *arXiv preprint arXiv:cs/0504107*.

Arman, A. and Tsaturian, S. (2017). The maximum number of cycles in a graph with fixed number of edges. *arXiv preprint arXiv:1702.02662*.

Babel, L. (1991). Finding maximum cliques in arbitrary and in special graphs. *Computing*, 46(4): 321–341.

Bagnara, R., Hill, P. M., Zaffanella, E., and Bagnara, A. (2016). *Parma Polyhedra Library 1.2 User's Manual*. Samurai Media Limited, United Kingdom.

Baïou, M. and Barahona, F. (2008). On the p-median polytope of Y-free graphs. *Discrete Optimization*, 5(2): 205–219.

Baïou, M. and Barahona, F. (2011). On the linear relaxation of the p-median problem. *Discrete Optimization*, 8(2): 344–375.

Baïou, M. and Barahona, F. (2016). On the p-median polytope and the directed odd cycle inequalities: Triangle-free oriented graphs. *Discrete Optimization*, 22: 206–224.

Baïou, M., Beaudou, L., Li, Z., and Limouzy, V. (2013). Hardness and algorithms for variants of line graphs of directed graphs. In *International Symposium on Algorithms and Computation*, 196–206. Springer.

Barahona, F. and Mahjoub, A. R. (1986). On the cut polytope. *Mathematical programming*, 36(2): 157–173.

Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. (1999). The maximum clique problem. In *Handbook of Combinatorial Optimization*, 1–74. Springer.

Bonami, P., Günlük, O., and Linderoth, J. (2018). Globally solving nonconvex quadratic programming problems with box constraints via integer programming methods. *Mathematical Programming Computation*, 10(3): 333–382.

Boros, E., Crama, Y., and Hammer, P. L. (1992). Chvátal cuts and odd cycle inequalities in quadratic 0–1 optimization. *SIAM Journal on Discrete Mathematics*, 5(2): 163–177.

Bron, C. and Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9): 575–577.

Burer, S. (2010). Optimizing a polyhedral-semidefinite relaxation of completely positive programs. *Mathematical Programming Computation*, 2(1): 1–19.

Burer, S. and Vandenbussche, D. (2009). Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. *Computational Optimization and Applications*, 43(2): 181–195.

Caprara, A. and Fischetti, M. (1996). {0, 1/2}-Chvátal-Gomory cuts. *Mathematical Programming*, 74(3): 221–235.

Carr, R. D. and Lancia, G. (2002). Compact vs. exponential-size LP relaxations. *Operations Research Letters*, 30(1): 57–65.

Carraghan, R. and Pardalos, P. M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9(6): 375–382.

Cheng, E. (1998). Separating subdivision of bicycle wheel inequalities over cut polytopes. *Operations Research Letters*, 23(1-2): 13–19.

Cheng, E. and Cunningham, W. H. (1997). Wheel inequalities for stable set polytopes. *Mathematical Programming*, 77(2): 389–421.

Cheng, E. and de Vries, S. (2002). Antiweb-wheel inequalities and their separation problems over the stable set polytopes. *Mathematical Programming*, 92(1): 153–175.

Chvátal, V. (1975). On certain polytopes associated with graphs. *Journal of Combinatorial Theory, Series B*, 18(2): 138–154.

Conforti, M., Cornuéjols, G., and Zambelli, G. (2010). Extended formulations in combinatorial optimization. *4OR*, 8(1): 1–48.

Coniglio, S. and Gualandi, S. (2014). On the exact separation of rank inequalities for the maximum stable set problem. `http://www.optimization-online.org/DB_HTML/2014/08/4514.html`. Accessed: 2020-05-11.

de Vries, S. (2015). Faster separation of 1-wheel inequalities by graph products. *Discrete Applied Mathematics*, 195: 74–83.

de Vries, S., Friedrich, U., and Perscheid, B. (2019). An extended formulation for the 1-wheel inequalities of the stable set polytope. *Networks*, 75(1): 86–94.

de Vries, S. and Perscheid, B. (2019). Tight compact extended relaxations for nonconvex quadratic programming problems with box constraints. `http:///www.optimization-online.org/DB_HTML/2019/09/7360.html`. Accessed: 2020-05-11.

de Vries, S. and Perscheid, B. (2020). A smaller extended formulation for the odd cycle inequalities of the stable set polytope. `http://www.optimization-online.org/DB_HTML/2019/09/7365.html`. Accessed: 2020-05-11.

Diestel, R. (2010). *Graphentheorie*. Springer.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1): 269–271.

Even, S. (2012). *Graph Algorithms*. Cambridge University Press.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to NP-completeness*. WH Freeman and Company, San Francisco.

Gerards, A. M. and Schrijver, A. (1986). Matrices with the Edmonds–Johnson property. *Combinatorica*, 6(4): 365–379.

Giandomenico, M. and Letchford, A. N. (2006). Exploring the relationship between max-cut and stable set relaxations. *Mathematical Programming*, 106(1): 159–175.

Giandomenico, M., Letchford, A. N., Rossi, F., and Smriglio, S. (2009). An application of the Lovász–Schrijver M(K,K) operator to the stable set problem. *Mathematical Programming*, 120(2): 381–401.

Giandomenico, M., Rossi, F., and Smriglio, S. (2013). Strong lift-and-project cutting planes for the stable set problem. *Mathematical Programming*, 141(1-2): 165–192.

Goemans, M. X. and Myung, Y.-S. (1993). A catalog of Steiner tree formulations. *Networks*, 23(1): 19–28.

Gollub, M. G., Dubé, R., Sommer, H., Gilitschenski, I., and Siegwart, R. (2017). A partitioned approach for efficient graph–based place recognition. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst./Workshop Planning, Perception Navigat. Intell. Veh.*

Golumbic, M. C. (2004). *Algorithmic Graph Theory and Perfect Graphs*. Elsevier.

Gross, J. L. and Yellen, J. (2004). *Handbook of Graph Theory*. CRC press.

Gross, J. L. and Yellen, J. (2005). *Graph Theory and its Applications*. CRC press.

Grötschel, M., Lovász, L., and Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2): 169–197.

Grötschel, M., Lovász, L., and Schrijver, A. (1988). *Geometric Algorithms and Combinatorial Optimization*. Springer.

Grötschel, M. and Pulleyblank, W. R. (1981). Weakly bipartite graphs and the max-cut problem. *Operations Research Letters*, 1(1): 23–27.

Hagberg, A., Swart, P., and Schult, D. (2008). Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Laboratory, New Mexico. `https://www.osti.gov/biblio/960616`. Accessed: 2020-05-11.

Hammack, R., Imrich, W., and Klavžar, S. (2011). *Handbook of Product Graphs*. CRC press.

Hoffman, K. L. and Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6): 657–682.

Husfeldt, T. (2015). Graph colouring algorithms. *arXiv preprint arXiv:1505.05825*.

Johnson, D. J. and Trick, M. A. (1996). *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, USA.

Johnson, D. W. (1974). Worst case behavior of graph coloring algorithms. In *Proceedings of the 5th Southeast Conference on Combinatorics, Graph Theory, and Computing*, 513–527.

Jungnickel, D. (2013). *Graphs, Networks, and Algorithms*. Springer.

Kaibel, V. (2011). Extended formulations in combinatorial optimization. *arXiv preprint arXiv:1104.1023*.

Kaibel, V., Lee, J., Walter, M., and Weltge, S. (2016). Extended formulations for independence polytopes of regular matroids. *Graphs and Combinatorics*, 32(5): 1931–1944.

Kariv, O. and Hakimi, S. L. (1979). An algorithmic approach to network location problems. I: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3): 513–538.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, 85–103. Springer.

Konc, J. and Janežič, D. (2007). An improved branch and bound algorithm for the maximum clique problem. *MATCH Communications in Mathematical and in Computer Chemistry*, 58: 569–590.

Koster, A. M., Zymolka, A., and Kutschka, M. (2009). Algorithms to separate {0, 1/2}-Chvátal-Gomory cuts. *Algorithmica*, 55(2): 375–391.

Lancia, G. and Serafini, P. (2011). An effective compact formulation of the max cut problem on sparse graphs. *Electronic Notes in Discrete Mathematics*, 37: 111–116.

Lancia, G. and Serafini, P. (2014). Deriving compact extended formulations via LP-based separation techniques. *4OR*, 12(3): 201–234.

Lovász, L. and Schrijver, A. (1991). Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, 1(2): 166–190.

Martin, R. K. (1991). Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters*, 10(3): 119–128.

Matula, D. W., Marble, G., and Isaacson, J. D. (1972). Graph coloring algorithms. In *Graph Theory and Computing*, 109–122. Elsevier.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems. *Mathematical Programming*, 10(1): 147–175.

Meghanathan, N. (2015). Distribution of maximal clique size of the vertices for theoretical small–world networks and real–world networks. *arXiv preprint arXiv:1508.01668*.

Meghanathan, N. (2016). Correlation analysis between maximal clique size and centrality metrics for random networks and scale-free networks. *Computer and Information Science*, 9(2): 41–57.

Mitchem, J. (1976). On various algorithms for estimating the chromatic number of a graph. *The Computer Journal*, 19(2): 182–183.

Moon, J. W. and Moser, L. (1965). On cliques in graphs. *Israel Journal of Mathematics*, 3(1): 23–28.

Nemhauser, G. L. and Sigismondi, G. (1992). A strong cutting plane/branch-and-bound algorithm for node packing. *Journal of the Operational Research Society*, 43(5): 443–457.

Nemhauser, G. L. and Trotter, L. E. (1974). Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6(1): 48–61.

Östergård, P. R. J. (2002). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1-3): 197–207.

Padberg, M. W. (1973). On the facial structure of set packing polyhedra. *Mathematical Programming*, 5(1): 199–215.

Pattabiraman, B., Patwary, M. M. A., Gebremedhin, A. H., Liao, W.-k., and Choudhary, A. (2013). Fast algorithms for the maximum clique problem on massive sparse graphs. In *International Workshop on Algorithms and Models for the Web-Graph*, 156–169. Springer.

Prosser, P. (2012). Exact algorithms for maximum clique: A computational study. *Algorithms*, 5(4): 545–587.

Rebennack, S., Oswald, M., Theis, D. O., Seitz, H., Reinelt, G., and Pardalos, P. M. (2011). A branch and cut solver for the maximum stable set problem. *Journal of Combinatorial Optimization*, 21(4): 434–457.

Reese, J. (2006). Solution methods for the p-median problem: An annotated bibliography. *Networks*, 48(3): 125–142.

Rosgen, B. and Stewart, L. (2007). Complexity results on graphs with few cliques. *Discrete Mathematics and Theoretical Computer Science*, 9(1).

Rossi, F. and Smriglio, S. (2001). A branch-and-cut algorithm for the maximum cardinality stable set problem. *Operations Research Letters*, 28(2): 63–74.

Rossi, R. A., Gleich, D. F., Gebremedhin, A. H., and Patwary, M. M. A. (2014). Fast maximum clique algorithms for large graphs. In *Proceedings of the 23rd International Conference on World Wide Web*, 365–366.

San Segundo, P., Lopez, A., Artieda, J., and Pardalos, P. M. (2017). A parallel maximum clique algorithm for large and massive sparse graphs. *Optimization Letters*, 11(2): 343–358.

San Segundo, P., Lopez, A., and Pardalos, P. M. (2016). A new exact maximum clique algorithm for large and massive sparse graphs. *Computers & Operations Research*, 66: 81–94.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*. John Wiley & Sons.

Schrijver, A. (2003). *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer Science & Business Media.

Seidman, S. B. (1983). Network structure and minimum degree. *Social Networks*, 5(3): 269–287.

Tomita, E. and Seki, T. (2003). An efficient branch-and-bound algorithm for finding a maximum clique. In *International Conference on Discrete Mathematics and Theoretical Computer Science*, 278–289. Springer.

Trotter, L. E. (1975). A class of facet producing graphs for vertex packing polyhedra. *Discrete Mathematics*, 12(4): 373–388.

Tutte, W. T. (2001). *Graph Theory, Reprint of the 1984 Original*, volume 21. Cambridge University Press.

Vandenbussche, D. and Nemhauser, G. (2005). A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102(3): 559–575.

Verma, A., Buchanan, A., and Butenko, S. (2015). Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on Computing*, 27(1): 164–177.

Walteros, J. L. and Buchanan, A. (2018). Why is maximum clique often easy in practice? `http://www.optimization-online.org/DB_HTML/2018/07/6710.html`. Accessed: 2020-05-11.

Yannakakis, M. (1991). Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43: 441–466.

Yu, T. and Liu, M. (2017). A linear time algorithm for maximal clique enumeration in large sparse graphs. *Information Processing Letters*, 125: 35–40.

Ziegler, G. M. (2007). *Lectures on Polytopes, Updated Seventh Printing*. Springer.