OPTIMIZATION METHODS FOR CLUSTER ANALYSIS IN NETWORK-BASED DATA MINING

A Dissertation

by

SEYEDMOHAMMADHOSSEIN HOSSEINIAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2021

Major Subject: Industrial Engineering

ABSTRACT

This dissertation focuses on two optimization problems that arise in network-based data mining, concerning identification of basic community structures (clusters) in graphs: the maximum edge weight clique and maximum induced cluster subgraph problems. We propose a continuous quadratic formulation for the maximum edge weight clique problem, and establish the correspondence between its local optima and maximal cliques in the graph. Subsequently, we present a combinatorial branch-and-bound algorithm for this problem that takes advantage of a polynomial-time solvable nonconvex relaxation of the proposed formulation. We also introduce a linear-time-computable analytic upper bound on the clique number of a graph, as well as a new method of upper-bounding the maximum edge weight clique problem, which leads to another exact algorithm for this problem. For the maximum induced cluster subgraph problem, we present the results of a comprehensive polyhedral analysis. We derive several families of facet-defining valid inequalities for the IUC polytope associated with a graph. We also provide a complete description of this polytope for some special classes of graphs. We establish computational complexity of the separation problems for most of the considered families of valid inequalities, and explore the effectiveness of employing the corresponding cutting planes in an integer (linear) programming framework for the maximum induced cluster subgraph problem.

DEDICATION

To my parents.

ACKNOWLEDGMENTS

I am extremely grateful to my advisor, Dr. Sergiy Butenko. I learned a lot from Sergiy and enjoyed every day of working with him. I am also very thankful to my advisory committee members, Drs. Lewis Ntaimo, Kiavash Kianfar, and Jianer Chen, and the Associate Department Head for Graduate Affairs, Dr. Alfredo Garcia, for their time and support. I would also like to pay my gratitude to Drs. Kenneth Reinschmidt, recently deceased, and Warren D. Reece, for their encouragement and advice when I was changing my academic path. Finally, my highest appreciation goes to my wife, Mona, without whose understanding and love completing this work would have been impossible.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors Sergiy Butenko (advisor), Lewis Ntaimo, and Kiavash Kianfar of the Department of Industrial and Systems Engineering, and Professor Jianer Chen of the Department of Computer Science and Engineering.

All work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by graduate assistantship from the Department of Industrial and Systems Engineering of Texas A&M University and Dr. Sergiy Butenko, and a graduate teaching fellowship from the College of Engineering of Texas A&M University.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Graph theory proves to be a powerful tool in analysis of complex systems with numerous applications in science and engineering. Many real-life systems can be described as a set of components that hold *pairwise* relationships while complex properties of the system can be explained in terms of these simple connections. Graphs are mathematical abstractions of such systems, where the system components are represented by vertices (nodes) and the pairwise relationships among them by edges (links). Graph theory provides analytical means to study properties of these systems, and presents algorithmic solutions to the corresponding problems.

In network-based system analysis, *communities* are of crucial importance, and identification of basic community structures (clusters) in a graph—constructed based on data collected from a real-life system—is considered a fundamental task in network-based data mining. This task gives rise to the concept of a *perfect cluster*, which in graph-theoretic terminology is referred to as *clique*. The term clique was first used in the context of social network analysis to refer to a group of people who are all mutual friends of one another [1]. Consider a graph whose vertices represent individuals in a social network; two vertices of the graph are connected by an edge if the corresponding individuals are friends. Then, a clique is a subset of pairwise-connected (adjacent) vertices. This model finds numerous applications in a wide range of disciplines, and is regarded as one of the most fundamental concepts of graph theory. In this regard, over the past few decades, a great deal of research has been dedicated to clique and many problems that center around it.

In this dissertation, we study two optimization problems that stem from this model in cluster analysis of graphs. The first problem is the *maximum edge weight clique problem*, which is to find a clique with the maximum weight of the corresponding induced subgraph in an edge-weighted graph. An edge-weighted graph is a graph whose edges are associated with numerical parameters (weights). The weights represent additional information about the underlying system, and the problems defined on edge-weighted graphs aim to provide a more accurate analysis of the system by exploiting this advantage. As a case in point, consider a graph representing similarities among data objects. That is, the vertices corresponding to two objects are connected by an edge if they are considered similar. Then, a clique represents a set of objects that are all similar to one another. This construction, however, ignores the fact that similarity is often not a binary quality but a continuum, which is quantified by some measure; some links in the network are stronger/weaker than the others, and the cohesion within subgroups depends on the strength of the links. Therefore, in many applications, it is necessary to include the similarity measure as the weight of the edges in the graph, and look for cliques that exhibit strong relationships according to the measure. Chapters 2 and 3 of this dissertation are dedicated to the maximum edge weight clique problem.

The second problem is the *maximum induced cluster subgraph problem*, which is also referred to as the *maximum independent union of cliques problem* in the literature. This problem is to find a maximum-cardinality subset of vertices such that every connected component of the corresponding induced subgraph is a complete graph, i.e., its vertex set forms a clique. Such an induced subgraph provides important information about the heterogeneous structure of the graph, including the inherent number of its clusters and their centroids, which are essential to graph clustering methods. Chapter 4 of this dissertation is dedicated to the maximum induced cluster subgraph problem.

1.1 Summary of Contributions

In Chapter 2, we present a continuous quadratic formulation for the maximum edge weight clique (MEWC) problem, and study its characteristics in terms of local and global optimality. In particular, we establish the correspondence between local optima of the proposed formulation and maximal cliques in the graph. Based on this result, we also present an exact algorithm to solve the MEWC problem. The algorithm is a combinatorial branch-and-bound procedure that takes advantage of an algebraic upper bound and a construction heuristic based on the proposed quadratic formulation.

In Chapter 3, we investigate the connection between the classical maximum (cardinality) clique and MEWC problems. We derive an analytic upper bound on the clique number of a graph from a Lagrangian relaxation of an integer (linear) programming formulation of the MEWC problem. Furthermore, we utilize coloring-based bounds on the clique number of a graph in a novel upper-bounding scheme for the MEWC problem. This scheme is employed within a combinatorial branch-and-bound framework, yielding another exact algorithm for this problem.

In Chapter 4, we provide a comprehensive study of the maximum induced cluster subgraph problem from the viewpoint of polyhedral combinatorics. We present several families of facet-inducing valid inequalities for the independent-union-of-cliques (IUC) polytope associated with a graph. For some special classes of graphs, we provide a full description of this polytope. We also study computational complexity of the separation problems for the identified families of valid inequalities, and establish their effectiveness in the integer (linear) programming solution methods of the maximum induced cluster subgraph problem.

In Chapter 5, we conclude this dissertation by a brief summary and potential directions for future research.

2. A NONCONVEX QUADRATIC OPTIMIZATION APPROACH TO THE MAXIMUM EDGE WEIGHT CLIQUE PROBLEM*

2.1 Introduction

Given a simple, undirected graph $G = (V, E)$, where $V = \{1, \dots, n\}$ is the set of vertices and E is the set of edges, a *clique* is a subset of vertices $C \subseteq V$ inducing a complete subgraph. A clique is called *maximal* if it is not a (proper) subset of a larger clique, and *maximum* if there is no lager clique in the graph. The *maximum clique problem* is to find a clique of maximum cardinality in G. The cardinality of a maximum clique in G is called the *clique number* of the graph, and is denoted by $\omega(G)$.

The *maximum edge weight clique (MEWC) problem* is a generalization of the maximum clique problem to edge-weighted graphs, which seeks a clique C with the maximum total weight of the edges in the corresponding induced subgraph $G[C]$. More formally, given a weight w_{ij} associated with each edge $\{i, j\} \in E$, the weight of a clique C is defined as $W(C) = \sum_{\{i,j\} \in E(C)} w_{ij}$, and the MEWC problem is to find a clique C in G that maximizes $W(C)$. If all edge weights of G are equal to 1, finding a maximum edge weight clique is equivalent to finding a maximum clique C^* in G with $W(C^*) = \binom{\omega(G)}{2}$. Therefore, the MEWC problem is at least as difficult to solve as the maximum clique problem, which is known to be NP-hard [3]. The MEWC problem has a wide range of applications, including computer vision and pattern recognition [4, 5], marketing [6], bioinformatics [7, 8], and healthcare [9].

Most of the works on the edge-weighted cliques in the literature deal with complete input graphs, and look for a maximum edge weight clique of a cardinality not exceeding a given bound. The corresponding problem is referred to as the *maximum diversity problem*, and is formally stated as follows: given a complete (undirected) edge-weighted graph $G = (V, E)$, find a subset of vertices of cardinality at most $k \leq |V|$ with the maximum total weight of the interconnecting

* This chapter is reprinted with permission from "A nonconvex quadratic optimization approach to the maximum edge weight clique problem" by S. Hosseinian, D. B. M. M. Fontes, and S. Butenko [2]. Copyright © 2018, Springer Science+Business Media, LLC, part of Springer Nature.

edges. An instance of the MEWC problem can be transformed into an instance of the maximum diversity problem by adding dummy edges with sufficiently large negative weights, and then solved for $k = |V|$. The exact solution methods proposed for this problem mainly involve branch-and-cut algorithms based on integer (linear) programming formulations; see for example [10, 11, 12, 13, 14]. Several heuristic and metaheuristic methods have also been applied to the maximum diversity problem, including tabu search [15, 16, 17], memetic search [18], scatter search [19], and greedy randomized adaptive search procedure [20, 21]. Reviews of these methods can be found in [17, 22, 23].

The only previous works dealing with the MEWC problem that we are aware of are [24] and [25], as well as our recent papers [26] and [27], the first of which is a preliminary workshop version of the present work and the second – a survey paper focusing on mathematical optimization formulations and existing solution approaches. More specifically, Pullan [24] developed a phased local search heuristic for this problem, and Gouveia and Martins [25] presented a set of integer programming formulations by introducing new valid inequalities to classical formulations based on sparseness of the graph. Here, we also present an exact method to solve this problem. We introduce a quadratic formulation for the MEWC problem; we use a relaxation of the new formulation to draw an upper bound for this problem, and use this bound within a combinatorial branch-and-bound procedure. Our method also benefits from an efficient construction heuristic algorithm, which is derived from the proposed formulation.

Our approach is based on a continuous characterization of the MEWC problem. A connection between cliques and a continuous optimization problem was first established by Motzkin and Straus [28], who showed a correspondence between maximum cliques in a graph and optima of a certain standard quadratic program. Pardalos and Phillips [29] were the first to use this connection in developing a global optimization algorithm for the maximum clique problem. The Motzkin-Straus formulation and its generalizations have been extensively studied in [30, 31, 32, 33, 34, 35, 36, 37]. In particular, Bomze [33] introduced its regularized version, which ensures one-to-one correspondence between local maxima of the quadratic program and maximal cliques of the underlying graph. In a similar manner, we introduce a quadratic programming formulation for the MEWC problem, and present the relation between the continuous problem and the underlying graph in terms of global and local optima. Unlike the Motzkin-Straus formulation, which has the standard simplex as its feasible region, our formulation maximizes a quadratic function over a unit hypercube. Formulations of the *maximum independent set problem*, which is equivalent to the maximum clique problem in the complement graph, as a problem of maximizing a nonlinear function over a unit hypercube have been previously considered in [38, 39, 40, 41, 42, 43].

Throughout this chapter, we assume G is a *proper* graph and all its edge weights are positive. A graph G is a proper graph if it is not complete and does not have an isolated vertex. By focusing on proper graphs, we exclude some trivial cases with respect to the MEWC problem. We denote vectors by boldface lowercase, and matrices by boldface uppercase letters. By **0** and **1**, we denote the vectors of all zero and one, respectively. All vectors are column vectors, and $\|\cdot\|$ denotes the Euclidean norm.

2.2 Quadratic Programming Formulation for the MEWC Problem

We consider a simple, undirected, and edge-weighted graph $G = (V, E)$, where $|V| = n$ and the edge weights are given by $w_{ij},\ \forall\{i, j\} \in E$. For a vertex $i \in V$, let $N(i)$ denote the neighborhood of i in G, defined as $N(i) = \{j \in V \mid \{i, j\} \in E\}$. The closed neighborhood of i is $N[i] = N(i) \cup \{i\}$. Let $W^*(G)$ denote the edge weight of a MEWC in G. The *characteristic vector* of a subset of vertices $C \subseteq V$ is an n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$ such that $x_i = 1,\ \forall i \in C$, and $x_i = 0,\ \forall i \notin C$.

Proposition 1. *The MEWC problem on G can be formulated as the following quadratic program (QP):*

$$W^*(G) = \max_{\mathbf{x}\in[0,1]^n} \left(\sum_{\{i,j\}\in E} w_{ij}x_i x_j - \sum_{\{i,j\}\notin E} \bar{w}_{ij}x_i x_j \right), \tag{2.1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$ is the vector of variables corresponding to the vertices of G and $\bar{w}_{ij}$

is defined as follows:

$$\bar{w}_{ij} = \max\{\sum_{k\in N(i)} w_{ik}, \sum_{k\in N(j)} w_{jk}\} + \xi, \; \forall\{i,j\} \notin E, i \neq j, \tag{2.2}$$

for an arbitrarily small $\xi > 0$. Any global optimal solution of this QP is the characteristic vector of a MEWC in G.

Let $\mathbf{Q}$ be a square matrix of order n with the following structure:

$$\mathbf{Q}(i,j) = \begin{cases} 0, & i = j, \\ w_{ij}, & \{i,j\} \in E, \\ -\bar{w}_{ij}, & \{i,j\} \notin E; \end{cases} \tag{2.3}$$

then, (2.1) is equivalent to

$$\max_{\mathbf{x}\in[0,1]^n} \quad f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\,\mathbf{x}. \tag{\textbf{P}}$$

Note that $\mathbf{Q}$ is symmetric and it is always indefinite by definition of a proper graph.

Proposition 1 is easy to establish directly, but it will also follow from the results characterizing the connection between local maxima of (**P**) and maximal cliques in G in the next section. First, we present the standard optimality conditions for (**P**).

2.2.1 First order necessary conditions (FONC)

Let $\mathbf{x}^* \in [0,1]^n$ be a local maximum point of (**P**). Then, there are two non-negative vectors $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{\lambda} \in \mathbb{R}^n$ such that:

$$\mathbf{Q}\,\mathbf{x}^* = \boldsymbol{\lambda} - \boldsymbol{\mu} \quad (\boldsymbol{\lambda} \geq \mathbf{0},\; \boldsymbol{\mu} \geq \mathbf{0}), \tag{2.4}$$

$$\mu_i\, x_i^* = 0 \quad \text{and} \quad \lambda_i\,(1 - x_i^*) = 0,\; \forall i \in \{1,2,...,n\}, \tag{2.5}$$

where μ_i, λ_i and x_i^* are the i-th components of vectors $\boldsymbol{\mu}$, $\boldsymbol{\lambda}$ and $\mathbf{x}^*$, respectively. The condition (2.4) enforces dual feasibility, and (2.5) is the complementary slackness condition.

2.2.2 Second order necessary conditions (SONC)

Suppose $\mathbf{x}^*$ is a local maximum point of (**P**). Let

$$Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \{i \mid (x_i^* = 0 \text{ and } \mu_i > 0) \ \text{ or } \ (x_i^* = 1 \ \text{ and } \lambda_i > 0)\}, \tag{2.6}$$

then, in addition to the first order necessary conditions, $\mathbf{x}^*$ also satisfies

$$\mathbf{y}^\top \mathbf{Q}\,\mathbf{y} \le 0, \ \forall \mathbf{y} \in \mathbb{R}^n \quad \text{subject to} \quad y_i = 0, \ \forall i \in Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}). \tag{2.7}$$

2.2.3 Second order sufficient conditions (SOSC)

If a point $\mathbf{x}^*$ satisfies FONC, SONC, and

$$\mathbf{y}^\top \mathbf{Q}\,\mathbf{y} < 0, \ \forall \mathbf{y} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \quad \text{subject to} \quad y_i = 0, \ \forall i \in Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}), \tag{2.8}$$

then $\mathbf{x}^*$ is a strict local maximizer of (**P**).

2.3 Optimality Characterizations

Lemma 2. *Every local maximizer* $\mathbf{x}^*$ *of* (**P**) *is a binary vector, i.e.,* $\mathbf{x}^* \in \{0,1\}^n$*, with at least two components equal to 1.*

Proof. Suppose there exists $i \in \{1, 2, \dots, n\}$ such that $0 < x_i^* < 1$. By (2.5), $\mu_i = \lambda_i = 0$ must hold for this solution. By (2.4), this implies that

$$\mathbf{Q}^i\,\mathbf{x}^* = 0, \tag{2.9}$$

where $\mathbf{Q}^i$ denotes the i-th row of matrix $\mathbf{Q}$. Regarding the structure of matrix $\mathbf{Q}$, equation (2.9) can be written as

$$\sum_{j \in N(i)} w_{ij}\, x_j^* \ - \sum_{k \notin N(i)} \bar{w}_{ik}\, x_k^* \ + \ 0\, x_i^* = 0. \tag{2.10}$$

In order for $\mathbf{x}^*$ to satisfy (2.10), exactly one of the following conditions must hold at this point:

$$x_j^* = x_k^* = 0, \; \forall j, k \in \{1, 2, ..., n\} \backslash \{i\}, \tag{2.11}$$

$$\exists\, j \in N(i) \;\; s.t. \;\; x_j^* \neq 0 \quad \text{and} \quad \exists\, k \notin N(i) \;\; s.t. \;\; x_k^* \neq 0. \tag{2.12}$$

In the above notation, i, j, and k are distinct indices. Now, we show that either of the aforementioned cases leads to a contradiction, implying that such a point $\mathbf{x}^*$ does not exist. Consider the former case. Since $\mathbf{x}^*$ is a local maximum point of (**P**), $f(\mathbf{x}^*) \geq f(\mathbf{x})$ for all feasible points $\mathbf{x} \in B_\epsilon(\mathbf{x}^*)$, where $B_\epsilon(\mathbf{x}^*) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}^*\| \leq \epsilon\}$ denotes an ϵ-neighborhood of $\mathbf{x}^*$ in $\mathbb{R}^n$. It is apparent that $f(\mathbf{x}^*) = 0$ because f is a homogeneous quadratic function. However, increasing the value of $x_j^* : j \in N(i)$ from 0 to ϵ (a single variable) results in a distinct feasible point $\hat{\mathbf{x}} \in B_\epsilon(\mathbf{x}^*)$ with $f(\hat{\mathbf{x}}) = w_{ij}\, x_i^*\, \epsilon > 0$ which contradicts the local optimality of $\mathbf{x}^*$. Note that by the problem assumption, i.e., there is no isolated vertex in G, it is guaranteed that such a vertex $j \in N(i)$ exists.

In the latter case, we use SONC to draw a contradiction. Let Y denote the set of all vectors $\mathbf{y}$ that are considered for SONC at the local maximum point $\mathbf{x}^*$. That is,

$$Y = \{\mathbf{y} \mid \mathbf{y} \in \mathbb{R}^n : y_i = 0, \; \forall i \in Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda})\}. \tag{2.13}$$

Consider the vector $\tilde{\mathbf{y}}$, only two components of which, $\tilde{y}_i$ and $\tilde{y}_k$, are non-zero. Then, $\tilde{\mathbf{y}} \in Y$ if x_i^* , $x_k^* \neq 0$ and x_i^* , $x_k^* \neq 1$. These conditions on x_i^* are already met by assumption, i.e., $0 < x_i^* < 1$. We pick vertex $k \notin N(i)$ such that $x_k^* \neq 0$. Note that, existence of such a vertex k is assumed under this case. Besides, $x_k^* < 1$ is enforced by the magnitude of $\bar{w}_{ik}$ and (2.10). We assign $\tilde{y}_i = 1$ and $\tilde{y}_k = -1$. By this assignment, $\tilde{\mathbf{y}} \in Y$ and we have

$$\tilde{\mathbf{y}}^\top \mathbf{Q}\, \tilde{\mathbf{y}} = 2\,(-\bar{w}_{ik})(1)(-1) = 2\,\bar{w}_{ik} > 0. \tag{2.14}$$

Equation (2.14) indicates violation of SONC, which contradicts the local optimality of $\mathbf{x}^*$. This proves that each local maximizer is a binary vector.

Since every variable in (**P**) corresponds to a vertex in the graph, every local maximum point of (**P**) is the characteristic vector of a subset of vertices in G. Next, we show that at least two components are equal to 1 in any local maximizer $\mathbf{x}^*$ of (**P**). Since f is a homogeneous quadratic function, $f(\mathbf{x}^*) = 0$ if less than two coordinates of $\mathbf{x}^*$ are nonzero. For **0**, as the characteristic vector of an empty set of vertices, observe that increasing the values of two variables x_i and x_j such that $\{i, j\} \in E$ from 0 to $\epsilon/\sqrt{2}$ results in a new point $\hat{\mathbf{x}} \in B_\epsilon(\mathbf{0})$ with $f(\hat{\mathbf{x}}) = \frac{1}{2} w_{ij}\, \epsilon^2 > 0$. Similarly, for the characteristic vector of a single-vertex set $\{i\}$, increasing the variable of an adjacent vertex j from 0 to ϵ increases the function value to $w_{ij}\, \epsilon > 0$. This completes the proof. □

Theorem 3. $\mathbf{x}^*$ *is a local maximizer of* (**P**) *if and only if it is the characteristic vector of a maximal clique in* G.

Proof. First, we show that if $\mathbf{x}^*$ is a local maximizer of (**P**), then it is the characteristic vector of a maximal clique in G. By Lemma 2, all components of $\mathbf{x}^*$ have to be 0 or 1. Let $V = S \cup T$ be a partition of vertices with respect to the components of $\mathbf{x}^*$ as follows:

$$S = \{j \in V \mid x_j^* = 0\} \quad \text{and} \quad T = \{i \in V \mid x_i^* = 1\}. \tag{2.15}$$

Note that a proper graph has at least three vertices and we have already shown that cardinality of T is at least two.

By (2.5),

$$\mu_i = 0 \quad \text{and} \quad \lambda_i \geq 0,\ \forall i \in T, \tag{2.16}$$

$$\lambda_j = 0 \quad \text{and} \quad \mu_j \geq 0,\ \forall j \in S. \tag{2.17}$$

Equations (2.16)-(2.17) along with (2.4) imply that

$$\mathbf{Q}^i\,\mathbf{x}^* = \lambda_i \geq 0,\ \forall i \in T, \tag{2.18}$$

$$\mathbf{Q}^j\,\mathbf{x}^* = -\mu_i \leq 0,\ \forall j \in S, \tag{2.19}$$

where $\mathbf{Q}^i$ and $\mathbf{Q}^j$ denote rows of matrix $\mathbf{Q}$ corresponding to vertices $i \in T$ and $j \in S$, respectively. Taking into account the structure of matrix $\mathbf{Q}$ and magnitude of its components, (2.18) implies that for every vertex $i \in T$, all vertices that are not adjacent to i must have variables equal to zero. That is,

$$i \in T, k \notin N[i] \ \Rightarrow \ k \in S, \tag{2.20}$$

which implies that T is a clique. Similarly, (2.19) implies that for every vertex $j \in S$, there must exist at least one vertex i with $x_i^* = 1$ such that i and j are not adjacent in G. That is,

$$\forall j \in S\ \exists\, i \in T : \{i, j\} \notin E, \tag{2.21}$$

so T is a maximal clique.

Now, we show that if $\mathbf{x}^*$ is the characteristic vector of a maximal clique C in G, then it is a local maximizer of (**P**). It is obvious that $\mathbf{x}^*$ is a feasible point of (**P**). We start by showing that $\mathbf{x}^*$ satisfies FONC and SONC. Let

$$\mu_i = 0 \quad \text{and} \quad \lambda_i = \mathbf{Q}^i\,\mathbf{x}^*,\ \forall i \in C, \tag{2.22}$$

$$\lambda_j = 0 \quad \text{and} \quad \mu_j = -\,\mathbf{Q}^j\,\mathbf{x}^*,\ \forall j \notin C. \tag{2.23}$$

This assignment satisfies (2.5). It also satisfies $\mathbf{Q}\mathbf{x}^* = \boldsymbol{\lambda} - \boldsymbol{\mu}$. Therefore, in order to prove $\mathbf{x}^*$ is a stationary point, it suffices to show that

$$\lambda_i \geq 0,\ \forall i \in C, \quad \text{and} \quad \mu_j \geq 0,\ \forall j \notin C. \tag{2.24}$$

Given the structure of matrix $\mathbf{Q}$ and $\mathbf{x}^*$ being the characteristic vector of C, we have

$$\mathbf{Q}^i\,\mathbf{x}^* = \sum_{k\in C\cap N(i)} w_{ik} - \sum_{l\in C\backslash N(i)} \bar{w}_{il},\ \forall i \in C. \tag{2.25}$$

Besides, C being a maximal clique with at least two vertices implies that

$$\forall i \in C:\ C\backslash N(i) = \emptyset \quad \text{and} \quad C\cap N(i) \neq \emptyset. \tag{2.26}$$

Equations (2.22) and (2.25)-(2.26) imply that

$$\lambda_i = \sum_{k\in C\cap N(i)} w_{ik} > 0,\ \forall i \in C. \tag{2.27}$$

Similarly,

$$\mathbf{Q}^j\,\mathbf{x}^* = \sum_{k\in C\cap N(j)} w_{jk} - \sum_{l\in C\backslash N(j)} \bar{w}_{jl},\ \forall j \notin C, \tag{2.28}$$

and by maximality of C,

$$C\backslash N(j) \neq \emptyset,\ \forall j \notin C. \tag{2.29}$$

Regarding the magnitude of $\bar{w}_{jl}$, (2.23) and (2.28)-(2.29) imply that

$$\mathbf{Q}^j\,\mathbf{x}^* < 0 \quad \therefore \quad \mu_j = -\,\mathbf{Q}^j\,\mathbf{x}^* > 0,\ \forall j \notin C. \tag{2.30}$$

Therefore, the assignment of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ according to (2.22) and (2.23) satisfies FONC, so $\mathbf{x}^*$ is a stationary point of (**P**). Furthermore, $Z(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}) = V$, $Y = \{\mathbf{0}\}$, so SONC and SOSC are satisfied, implying that $\mathbf{x}^*$ is a strict local maximum point of this problem. □

Corollary 4. *A clique C is an optimal solution to the MEWC problem if and only if its characteristic vector is a global maximizer of* (**P**).

Proof. The feasible region of (**P**) is a compact set, so $f(\mathbf{x})$ attains its global maximum in a local maximizer. By Theorem 3, every local maximizer of (**P**) is the characteristic vector of a maximal clique in G. The objective value of a local maximizer $\mathbf{x}_C$ of (**P**), corresponding to a maximal clique C in G, is

$$f(\mathbf{x}_C) = \sum_{\{i,j\}\in E} w_{ij}x_ix_j - \sum_{\{i,j\}\notin E} \bar{w}_{ij}x_ix_j = \sum_{\{i,j\}\in E(C)} w_{ij} = W(C), \tag{2.31}$$

where $E(C)$ is the set of edges of the induced subgraph $G[C]$. Therefore, a global maximizer of (**P**) is the characteristic vector of a maximal clique in G with maximum total weight. $\square$

As stated previously, the maximum clique problem can be considered as a special case of the MEWC problem. Hence, the aforementioned results hold for this problem too. This is presented through the following corollary.

Corollary 5. (The Maximum Clique Problem) *The clique number $\omega(G)$ of a proper, undirected, and unweighted graph $G = (V, E)$ can be found through solving the following quadratic programming problem:*

$$\binom{\omega(G)}{2} = \max_{\mathbf{x}\in[0,1]^n} \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\,\mathbf{x}\,, \tag{2.32}$$

where matrix $\mathbf{Q}$ is constructed according to (2.3)*, with $w_{ij} = 1$, $\forall\{i,j\} \in E$, and $\bar{w}_{ij} = \max\{d(i)\,,\, d(j)\} + 1$, $\forall\{i,j\} \notin E$; $d(v)$ denotes the degree of a vertex $v \in V$.*

2.4 Solving the MEWC Problem

The results in the last section establish the relation between (**P**) and the MEWC problem in terms of optimality characteristics. We use these results to develop a method for solving the MEWC problem. In this section, we first present a construction heuristic that is derived from a polynomial-time solvable approximation of (**P**). Then, we introduce an algebraic upper bound for this problem based on solving a quadratic relaxation of (**P**). Finally, we present our solution method, which is a combinatorial branch-and-bound (B&B) procedure that uses an initial lower bound provided by the heuristic algorithm, and applies the algebraic upper bound to prune the search tree.

2.4.1 Construction heuristic

Problem (**P**) concerns maximization of a nonconvex quadratic function subject to a set of linear constraints, which is NP-hard [44]. However, quadratic optimization subject to an ellipsoid constraint is polynomial-time solvable [45]. We approximate (**P**) by replacing its unit hypercube constraint with a unit hypersphere and examining stationary points of the new problem. Similar approaches have been successfully exploited before by replacing the standard simplex in the Motzkin-Straus formulation [31, 36] or the unit hypercube in a QP formulation over a box [42] with an ellipsoid constraint to develop a heuristic algorithm for the maximum clique/independent set problem.

The new problem is obtained as follows. Assume that a MEWC consists of k vertices. Then $\mathbf{1}^\top \mathbf{x}^* = k$, so by adding this constraint and changing the variables by dividing each variable by k, we obtain a problem equivalent to (**P**),

$$\begin{aligned} \max_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\,\mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in [0, 1/k]^n \\ & \mathbf{1}^\top \mathbf{x} = 1. \end{aligned} \tag{2.33}$$

Finally, the feasible region of (2.33) is approximated with a unit hypersphere:

$$\begin{aligned} \max_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\,\mathbf{x} \\ \text{s.t.} \quad & \mathbf{x}^\top \mathbf{x} = 1. \end{aligned} \tag{2.34}$$

Suppose $\mathbf{x}^*$ is a local optimizer of (2.34). Then, the first order optimality conditions indicate

$$\mathbf{Q}\,\mathbf{x}^* = 2\,\mu\,\mathbf{x}^*, \tag{2.35}$$

$$\|\mathbf{x}^*\|^2 = 1, \tag{2.36}$$

where μ is the Lagrangian multiplier of the constraint.

Equations (2.35) and (2.36) imply that every local optimizer $\mathbf{x}^*$ of (2.34) is a normalized eigenvector of matrix $\mathbf{Q}$ corresponding to an eigenvalue $\lambda = 2\,\mu$. A heuristic solution to the MEWC problem is constructed by considering all eigenvectors of matrix $\mathbf{Q}$, each treated as an approximation of the characteristic vector of a subset of vertices in G. Algorithm 1 presents the corresponding procedure.

Algorithm 1 Quadratic-based Construction Heuristic (QCH)

```
1: function QCH($G$)
2:     $\tilde{C} = \emptyset$ ; $\tilde{W} = 0$
3:     construct matrix $\mathbf{Q}$ according to (2.3)
4:     for each eigenvector $\mathbf{x}_e$ of $\mathbf{Q}$ do
5:         $C_d$ = EXTRACTCLIQUE($\mathbf{x}_e$, $decreasing$)
6:         if $W(C_d) > \tilde{W}$ then                    ▷ $W(C_d)$: weight of the clique $C_d$
7:             $\tilde{C} \leftarrow C_d$ ; $\tilde{W} \leftarrow W(C_d)$
8:         end if
9:         $C_i$ = EXTRACTCLIQUE($\mathbf{x}_e$, $increasing$)
10:        if $W(C_i) > \tilde{W}$ then                    ▷ $W(C_i)$: weight of the clique $C_i$
11:            $\tilde{C} \leftarrow C_i$ ; $\tilde{W} \leftarrow W(C_i)$
12:        end if
13:    end for
14:    return $(\tilde{C}, \tilde{W})$
15: end function
```

The QCH algorithm operates as follows: first, it calculates eigenvectors of matrix $\mathbf{Q}$. Each eigenvector $\mathbf{x}_e$ is treated as an approximation of the characteristic vector of a maximal clique. Mapping from a real-valued vector $\mathbf{x}_e$ to a maximal clique is done through the EXTRACTCLIQUE function, demonstrated in Algorithm 2. This function sorts the vertices based on their values in vector $\mathbf{x}_e$. A clique is initiated by including the vertex corresponding to the first element of the sorted list, and is expanded by appending other vertices according to their orders in the list. To make sure the mapping output is indeed a clique, a vertex is added to the clique only if it is adjacent to all vertices that are already in the clique. Sorting components of $\mathbf{x}_e$ in a decreasing (non-increasing) order follows the idea that the closer a value is to 1, the higher priority its vertex has to be included in the incumbent clique. Since $\mathbf{x}_e$ components have signed values, an analogous mapping can be considered by giving priority to vertices whose values in $\mathbf{x}_e$ are close to -1. This mapping corresponds to sorting the $\mathbf{x}_e$ components in an increasing (non-decreasing) order. The

QCH algorithm examines both mappings by calling the EXTRACTCLIQUE function twice with different ordering arguments. The algorithm outputs a clique with the highest weight among all extracted cliques.

Algorithm 2 Clique extraction

```
1: function EXTRACTCLIQUE($\mathbf{x}$, $order$)          ▷ order: decreasing or increasing
2:     sort components of $\mathbf{x}$ according to order   ▷ sorted array: $[x^{(1)}, x^{(2)}, \ldots, x^{(n)}]$
3:     $C = \{v[x^{(1)}]\}$                               ▷ $v[x^{(1)}]$: vertex corresponding to $x^{(1)}$
4:     for $j = 2$ to $n$ do
5:         if $v[x^{(j)}] \in \bigcap_{i \in C} N(i)$ then
6:             $C \leftarrow C \cup \{v[x^{(j)}]\}$
7:         end if
8:     end for
9:     return $C$
10: end function
```

The heuristic results can potentially improve when the QCH algorithm is implemented for the closed neighborhood of each vertex. That is, if a vertex i was known to be contained in the optimal solution, then the problem would be simplified to finding a maximum edge weight clique in $G[N[i]]$, the subgraph induced by i and its neighbors. While it takes longer to examine all such subgraphs, the heuristic algorithm is likely to generate better solutions. Algorithm 3 demonstrates utilizing the QCH algorithm in this setting. The output of the QCH (or QCH-N) algorithm provides an initial lower bound for the combinatorial B&B procedure. We show the quality of these solutions in Section 2.5 through computational experiments.

2.4.2 Quadratic relaxation bound

As it will be shown in detail in the next section, the combinatorial B&B algorithm traverses a search tree to find a clique with the maximum edge weight in the graph. At every node of the tree, it considers a subgraph induced by the union of an incumbent clique C and a list of vertices L, called *candidate list*. The candidate list is composed of the vertices in $V \backslash C$ that are adjacent to all vertices in C, so appending each of them to C would result in a larger and heavier clique. A node

Algorithm 3 QCH-N, QCH used for closed neighborhoods

```
 1: function QCH-N($G$)
 2:     $\tilde{C} = \emptyset$ ; $\tilde{W} = 0$
 3:     for each vertex $i \in V$ do
 4:         run QCH on $G[N[i]]$ to obtain $\tilde{C}_i$ and $W(\tilde{C}_i)$
 5:         if $W(\tilde{C}_i) > \tilde{W}$ then
 6:             $\tilde{C} \leftarrow \tilde{C}_i$ ; $\tilde{W} \leftarrow W(\tilde{C}_i)$
 7:         end if
 8:     end for
 9:     return $(\tilde{C}, \tilde{W})$
10: end function
```

of the tree is pruned, i.e., the corresponding subgraph is excluded from further investigation, if an upper bound on the maximum clique-weight in the subgraph is not greater than a known global lower bound. In this section, we present a new method to calculate such upper bounds based on solving a relaxation of (**P**).

Consider $G' = G[C \cup L]$, the subgraph induced by the union of a clique C and its candidate list L. Let $\mathbf{Q}'$ be the corresponding matrix to G' as defined in Section 2.2. We seek an upper bound on $W^*_{G'}$, the maximum weight of a clique in G'. By Proposition 1,

$$W^*_{G'} = \max_{\mathbf{x} \in [0,1]^{n'}} \; \frac{1}{2} \mathbf{x}^\top \mathbf{Q}' \, \mathbf{x}, \tag{2.37}$$

where $n' = |C \cup L|$. As every vertex in L is adjacent to all vertices of C, any maximal clique in G' must contain C and by Theorem 3,

$$x_i = 1, \; \forall i \in C, \tag{2.38}$$

in an optimal solution of (2.37). Therefore, the objective function of (2.37) can be expanded as follows:

$$f(\mathbf{x}) = W(C) + \sum_{i \in L} x_i \left(\sum_{j \in C} w_{ij} \right) + \sum_{i \in L} \sum_{j \in L} \mathbf{Q}'(i,j) \, x_i x_j. \tag{2.39}$$

Results of the last section also indicate that the rest of the variables in the optimal solution are at their bounds. That is,

$$x_i \in \{0,1\},\ \forall i \in L. \tag{2.40}$$

Let $\mathbf{q}$ denote the vector of coefficients in the second term of (2.39), then (2.37) can be written as

$$W^*_{G'} = W(C) + \max_{\mathbf{x}\in\{0,1\}^{|L|}} \left(\mathbf{q}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}'_L \mathbf{x}\right), \tag{2.41}$$

where $\mathbf{Q}'_L$ is the principal submatrix of $\mathbf{Q}'$ corresponding to vertices of L. Note that $\mathbf{Q}'_L$ is not constructed directly from the subgraph induced by L because the negative components of this matrix should account for the edges connecting vertices of L and C as well as interconnecting edges of L.

We calculate an upper bound on $W^*_{G'}$ through the following equation:

$$\mathcal{Z}_{G'} = W(C) + \begin{pmatrix} \mathcal{Z}_L = \max\limits_{\mathbf{x}\in\mathbb{R}^{|L|}} & \mathbf{q}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{Q}'_L \mathbf{x} \\ \text{s.t.} & (\mathbf{x}-\mathbf{b})^\top(\mathbf{x}-\mathbf{b}) = \frac{|L|}{4} \end{pmatrix}, \tag{2.42}$$

where $\mathbf{b} \in \mathbb{R}^{|L|}$ is a vector with all components equal to $\frac{1}{2}$. It is clear that the feasible region of the quadratic optimization problem in (2.42) contains all 0-1 vectors of size $|L|$, so it is a relaxation of (2.41) and $\mathcal{Z}_{G'} \geq W^*_{G'}$. Note that even if G' is not proper, i.e., L is a clique, (2.41) remains valid and $\mathcal{Z}_{G'}$ will be an upper bound on $W^*_{G'}$.

Let $\mathbf{y} = \mathbf{x} - \mathbf{b}$ and $\mathbf{d} = \mathbf{Q}'_L\mathbf{1} + 2\mathbf{q}$. Then,

$$\mathcal{Z}_L = \frac{1}{2}\left(\max_{\mathbf{y}^\top\mathbf{y}=|L|/4} \mathbf{d}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{Q}'_L \mathbf{y}\right) + \text{const.} \tag{2.43}$$

By eigen-decomposition of matrix $\mathbf{Q}'_L$ and a linear transformation to the eigenvector space, (2.43) can be written as

$$\mathcal{Z}_L = \frac{1}{2}\left(\max_{\mathbf{y}'^\top\mathbf{y}'=|L|/4} \mathbf{d}'^\top \mathbf{y}' + \mathbf{y}'^\top \mathbf{\Lambda} \mathbf{y}'\right) + \text{const.} \tag{2.44}$$

In (2.44), $\mathbf{\Lambda}$ is a diagonal matrix composed of eigenvalues of $\mathbf{Q}'_L$ and $\mathbf{y}' = \mathbf{E}^\top\mathbf{y}$ and $\mathbf{d}' = \mathbf{E}^\top\mathbf{d}$, where $\mathbf{E}$ is a square matrix whose columns are orthonormal eigenvectors of $\mathbf{Q}'_L$ with the same order as the corresponding eigenvalues in $\mathbf{\Lambda}$. Let μ denote the Lagrangian multiplier of the single hypersphere constraint in (2.44). Then, by the first order optimality conditions, every local maximizer of (2.44) is a solution to the following system of equations:

$$2(\mathbf{\Lambda} - \mu \mathbf{I})\mathbf{y}' + \mathbf{d}' = 0, \tag{2.45}$$

$$\|\mathbf{y}'\|^2 = \frac{|L|}{4}. \tag{2.46}$$

The set of all μ for which there exists a vector $\mathbf{y}'$ such that $(\mu, \mathbf{y}')$ satisfies (2.45)-(2.46) is called the *spectrum* of this system.

Forsythe and Golub [46] have studied optimization of a quadratic function subject to a single hypersphere constraint in detail. In particular, they have shown that the global maximizer of this problem corresponds to the greatest μ in the spectrum of (2.45)-(2.46). Let $\{\lambda_1, \lambda_2, \ldots, \lambda_{|L|}\}$ be the set of eigenvalues of $\mathbf{Q}'_L$ (possibly with multiplicity) and λ^{max} denote the maximum eigenvalue of this matrix. If there exists $i \in \{1, \ldots, |L|\}$ such that $\lambda_i = \lambda^{max}$ and $d'_i \neq 0$, then the greatest μ in the spectrum of (2.45)-(2.46) is given by the greatest root of the following univariate function, which we denote by μ^*, i.e., $\mu^* = \max\{\mu \mid \Phi(\mu) = 0\}$:

$$\Phi(\mu) = \sum_{i \in \mathcal{I}} \frac{(\lambda_i\, b'_i + q'_i)^2}{(\lambda_i - \mu)^2} - \frac{|L|}{4}\ , \ \mathcal{I} = \{i \mid \lambda_i\, b'_i + q'_i \neq 0\}, \tag{2.47}$$

where b'_i and q'_i are the i-th components of vectors $\mathbf{b}' = \mathbf{E}^\top \mathbf{b}$ and $\mathbf{q}' = \mathbf{E}^\top \mathbf{q}$, respectively. It is easy to see that $d'_i = \lambda_i\, b'_i + q'_i$, and under the above condition, μ^* always exists and belongs to the interval $(\lambda^{max}, +\infty)$. In this case, $\mathcal{Z}_L$ can be directly calculated as follows:

$$\mathcal{Z}_L \ = \sum_{i=1}^{|L|} \frac{(\mu^*\, b'_i + q'_i)\,[\lambda_i(\mu^*\, b'_i + q'_i) + 2\mu^* q'_i]}{2(\lambda_i - \mu^*)^2}. \tag{2.48}$$

Note that $\Phi(\mu)$ is monotonically decreasing in the interval $(\lambda^{max}, +\infty)$, so μ^* can be calculated by line search methods to a desired precision.

But, if $d'_i = 0$ for all $\lambda_i = \lambda^{max}$, then μ^* does not necessarily have the maximum value in the spectrum of (2.45)-(2.46). In this case, any eigenvalue $\lambda > \mu^*$ that satisfies (2.49) and (2.50) will generate a better objective value for this problem:

$$i \in \bar{\mathcal{I}},\ \forall \lambda_i = \lambda, \tag{2.49}$$

$$\sum_{i \notin \mathcal{J}_\lambda} \frac{(\lambda_i\, b'_i + q'_i)^2}{(\lambda_i - \lambda)^2} < \frac{|L|}{4}\,, \tag{2.50}$$

where $\bar{\mathcal{I}} = \{i \mid \lambda_i b'_i + q'_i = 0\}$ and $\mathcal{J}_\lambda = \{i \in \bar{\mathcal{I}} \mid \lambda_i = \lambda\}$. Let λ^* denote the largest eigenvalue of $\mathbf{Q}'_L$ that satisfies these conditions. Then, $\mathcal{Z}_L$ is given by

$$\mathcal{Z}_L \ = \frac{1}{2} \sum_{i=1}^{|L|} \lambda_i (y_i + b'_i)^2 + q'_i (y_i + b'_i), \tag{2.51}$$

where

$$y_i = \begin{cases} \dfrac{-(\lambda_i\, b'_i + q'_i)}{\lambda_i - \lambda^*}, & i \notin \mathcal{J}_{\lambda^*}, \\ \sqrt{\alpha}, & i = j \ \text{ for a single } \ j \in \mathcal{J}_{\lambda^*} :\ \alpha = \dfrac{|L|}{4} - \displaystyle\sum_{i \notin \mathcal{J}_{\lambda^*}} y_i^2, \\ 0, & i \in \mathcal{J}_{\lambda^*} \backslash \{j\}. \end{cases} \tag{2.52}$$

We use these results to calculate an upper bound for the MEWC problem on the subgraphs visited throughout the combinatorial B&B process.

2.4.3 Combinatorial branch-and-bound procedure

Our algorithm utilizes a classical combinatorial method for implicit enumeration of maximal cliques in a graph. This framework has been vastly used in the algorithms proposed for the maximum clique problem with different pruning criteria and implementation details; see for example [47, 48, 49, 50, 51, 52]. We adapt this method for the MEWC problem by applying the upper bound of the last section, and present a combinatorial B&B algorithm for this problem. Algorithm 4 demonstrates the outline of our method.

Algorithm 4 Combinatorial B&B algorithm w/ Quadratic relaxation bound (CBQ)

```
1: function CBQ($G, \tilde{C}, \tilde{W}$)          ▷ $\tilde{C}$: initial solution, $\tilde{W}$: initial lower bound
2:     $C = \emptyset$ ; $W = 0$
3:     $C^* = \tilde{C}$ ; $W^* = \tilde{W}$
4:     $L$ = INITIALIZATION($G$)
5:     BRANCH($G, L, C, C^*, W, W^*$)
6:     return ($C^*$, $W^*$)
7: end function
```

The CBQ algorithm starts with making a sorted array of vertices L by calling the INITIALIZATION function. The order of vertices in this array determines the sequence of visiting cliques in the graph. This initial ordering aims to keep the subproblems formed by the BRANCH function small. BRANCH is a recursive function that examines maximal cliques in a systematic manner. All input arguments of this function are global and updated through the search process.

The INITIALIZATION function stores the vertices in L such that $L(k)$, the k-th vertex in L, has the smallest degree in the subgraph induced by $\{L(1), L(2), \ldots, L(k)\}$. If there are two (or more) vertices with minimum degree in this subgraph, their relative positions in L are determined based on degrees of their neighbors. Let G^k denote the subgraph induced by the first k vertices in L. If more than one vertex in G^k have the minimum degree, then $L(k)$ is the vertex with the minimum number of neighbor-of-neighbors, denoted by σ in Algorithm 5, among them.

Algorithm 6 shows details of the BRANCH function. In every call, the inputs of this function are an incumbent clique C and its weight W, the best-known solution C^* and its weight W^*, and a list of candidate vertices to expand the incumbent clique, denoted by L. In fact, $L \subseteq V \backslash C$ contains vertices that are adjacent to all vertices of C. Initially, C is an empty set with its weight equal to zero, C^* and W^* are provided by the QCH heuristic solution, and the candidate list involves all vertices of the graph sorted according to the initial ordering.

The BRANCH function operates as follows: foremost, it calls the PRUNE function, which determines whether the subgraph induced by $C \cup L$ could contain a clique heavier than the best-known solution. If the answer is negative, then the corresponding subgraph is excluded from further in-

Algorithm 5 Initialization

```
function INITIALIZATION($G$)
    $L$ = an empty array of size $n$
    for $k = n$ to $1$ do
        $R \leftarrow$ the set of vertices with minimum degree in $G$
        if $|R|$ = 1 then
            $u \leftarrow$ the vertex in $R$
        else
            for every vertex $v \in R$ do
                $\sigma(v) = \sum_{i \in N(v)} degree(i)$
            end for
            $u \leftarrow$ a vertex in $R$ with minimum $\sigma$
        end if
        $L(k) \leftarrow u$
        $G \leftarrow G \backslash \{u\}$
    end for
    return $L$
end function
```

vestigation and the function returns. Otherwise, it picks the last vertex in L and appends it to C. Correspondingly, a new candidate list for the expanded clique, denoted by L_v in Algorithm 6, is formed. Then, the function calls itself on the subproblem defined by the new clique and the corresponding candidate list. If L_v is empty then the incumbent clique is maximal, thus its weight is compared against the best-known weight W^* and the best solution is kept. Since the BRANCH function traverses the search tree in a depth-first manner, the function returns only when it has investigated all possible outcomes of expanding C by appending $v \in L$. In other words, all cliques that contain $C \cup \{v\}$ have been examined upon solving the subproblem, so v is removed from the expanded clique as well as the candidate list.

Finally, presentation of the CBQ algorithm concludes by describing the pruning process. The PRUNE function determines whether or not a subproblem should be processed further in the BRANCH procedure. A subproblem is pruned if an upper bound on maximum weight of the cliques in the subgraph induced by $C \cup L$ is not greater than the best-known weight so far. We use the quadratic upper bound presented in the last section to prune the search tree. Algorithm 7 demon-

Algorithm 6 Branch-and-bound procedure

```
1: function BRANCH($G, L, C, C^*, W, W^*$)
2:     while $L \neq \emptyset$ do
3:         $p$ = PRUNE($G, L, C, W, W^*$)
4:         if $p = \texttt{true}$ then                                  ▷ the branch is pruned
5:             return
6:         else                                                         ▷ the branch is not pruned
7:             $v \leftarrow$ last vertex in $L$
8:             $\delta W = \sum_{i \in C} w_{iv}$
9:             $C \leftarrow C \cup \{v\}$
10:            $W \leftarrow W + \delta W$
11:            $L_v =$ an array of neighbors of $v$ in $L$ with the same relative order as in $L$
12:            if $L_v \neq \emptyset$ then
13:                BRANCH($G$, $L_v$, $C$, $C^*$, $W$, $W^*$)
14:            else if $W > W^*$ then
15:                $C^* \leftarrow C$ ; $W^* \leftarrow W$
16:            end if
17:            $C \leftarrow C \backslash \{v\}$
18:            $W \leftarrow W - \delta W$
19:        end if
20:        $L \leftarrow L \backslash \{v\}$
21:    end while
22:    return
23: end function
```

strates details of the pruning procedure.

Clearly, sum of all edge weights in the subgraph induced by $C \cup L$, denoted by $W(C \cup L)$ in Algorithm 7, is a trivial upper bound for the MEWC problem on this subgraph. Although the algebraic bound presented in the last section could be much tighter, it cannot be guaranteed to reach $W(C \cup L)$ all the time. In this regard, the algorithm always compares the algebraic bound $\mathcal{Z}_L$ with the combinatorial bound $W(C \cup L)$ and uses the tighter one in the pruning process. Besides, if $\lambda_i b'_i + q'_i = 0,\ \forall \lambda_i = \lambda^{max}$, we safely use $W(C \cup L)$ without calculating (2.51)-(2.52). We will show later, in the computational experiments section, that the frequency of using the combinatorial bound is negligible in comparison to that of the algebraic bound for most instances, implying effectiveness of the algebraic bound.

Algorithm 7 Pruning procedure

```
 1: function PRUNE($G, L, C, W, W^*$)
 2:     upper bound $= W(C \cup L) - W$
 3:     for $i = 1$ to $|L|$ do
 4:         $q_i = \sum_{j \in C} w_{ij}$
 5:     end for
 6:     construct matrix $\mathbf{Q}'_L$ according to (2.37)
 7:     for $i = 1$ to $|L|$ do
 8:         $\lambda_i$ = the $i$-th smallest eigenvalue of $\mathbf{Q}'_L$
 9:         $e^i$ = the $i$-th eigenvector of $\mathbf{Q}'_L$ corresponding to $\lambda_i$
10:     end for
11:     $\lambda^{max} \leftarrow \lambda_{|L|}$
12:     for $i = 1$ to $|L|$ do
13:         $q'_i = \sum_{j=1}^{|L|} e^i_j \, q_j$
14:         $b'_i = \frac{1}{2} \sum_{j=1}^{|L|} e^i_j$
15:     end for
16:     $i \leftarrow |L|$
17:     do
18:         if $\lambda_i b'_i + q'_i \neq 0$ then
19:             $\mu^*$ = root of $\Phi(\mu)$ according to (2.47) in the interval $(\lambda^{max}, +\infty)$
20:             $\mathcal{Z}_L = \sum_{i=1}^{|L|} \frac{(\mu^* b'_i + q'_i)\,[\lambda_i(\mu^* b'_i + q'_i) + 2\mu^* q'_i]}{2(\lambda_i - \mu^*)^2}$
21:             upper bound $\leftarrow$ min $\{\mathcal{Z}_L$, upper bound$\}$
22:             if upper bound $\leq W^* - W$ then
23:                 return true
24:             else
25:                 return false
26:             end if
27:         end if
28:         $i \leftarrow i - 1$
29:     while $\lambda_i = \lambda^{max}$
30:     if upper bound $\leq W^* - W$ then
31:         return true
32:     else
33:         return false
34:     end if
35: end function
```

2.5 Computational Experiments

In this section, we present computational results for the proposed approaches on 28 benchmark instances taken from DIMACS, RTN, and SC-NIP datasets. DIMACS instances are originally unweighted [53]. We are using an edge-weighted version of them that was first introduced in [24] for the MEWC problem. In these graphs, a positive weight is assigned to an edge $\{i, j\} \in E$ is given by $w_{ij} = (i+j) \bmod 200 + 1$. We refer to [25] for a description of RTN and SC-NIP graphs, and the associated edge weights. Table 2.1 presents characteristics of the test instances, where $|V|$, $|E|$ and "Density" denote the number of vertices, number of edges and edge density of the graph, respectively.

We compare our results with the ones of Gouveia and Martins [25] as their work is the most recent and competitive one on solving the MEWC problem using exact solution methods. They have used sparseness of the graph to improve classic integer programming formulations of this problem. Their work presents computational results of solving the MEWC problem through nine different integer programming formulations, namely F1, F11, F2, F21, F5, F52, F6, F61 and F62. We compare the performance of our algorithm with the best results among these models for each instance. In [25], the solution time for each instance and each model has been limited to 3 hours (10,800 seconds). We set the same upper time limit in our experiment. In the results to follow, we have used $\xi = 1$ in the calculation of $\bar{w}_{ij}$ according to (2.2).

Table 2.2 summarizes the main results in terms of solution time. In this table, W^* is the optimal weight for each instance (if known). The column CBQ presents the total CPU time in seconds taken by our algorithm to solve each instance on an Intel® Core i-7 CPU @2.90 GHz. The next column, titled "Gouveia & Martins", shows the best solution time among the nine integer programming formulations of [25] along with the best formulation in parenthesis. These are the times reported in [25] obtained using the ILOG/CPLEX 11.2 solver on an Intel® Core i-7 CPU @3.40 GHz. One of the most popular integer programming formulations of the MEWC problem is the one that was originally proposed in [11]. Gouveia and Martins have considered this formulation with one additional constraint and presented the corresponding computational results in their paper.

Table 2.1: Characteristics of the test instances (CBQ algorithm).

Name	$\|V\|$	$\|E\|$	Density	Name	$\|V\|$	$\|E\|$	Density
brock200-1	200	14,834	0.754	johnson8-2-4	28	210	0.556
brock200-2	200	9,876	0.496	johnson8-4-4	70	1,855	0.768
brock200-3	200	12,048	0.605	keller4	171	9,435	0.649
brock200-4	200	13,089	0.658	MANN-a9	45	918	0.927
C125.9	125	6,963	0.898	p-hat300-1	300	10,933	0.244
C250.9	250	27,984	0.899	p-hat300-2	300	21,928	0.489
c-fat200-1	200	1,534	0.077	p-hat300-3	300	33,390	0.744
c-fat200-2	200	3,235	0.163	d1-RTN	2,418	9,317	0.003
c-fat200-5	200	8,473	0.426	d3-RTN	4,755	26,943	0.002
hamming6-2	64	1,824	0.905	d7-RTN	6,511	44,615	0.002
hamming6-4	64	704	0.349	d15-RTN	7,965	62,136	0.002
hamming8-2	256	31,616	0.969	SC-NIP-m-t1	991	4,161	0.008
hamming8-4	256	20,864	0.639	SC-NIP-r-t1	1,393	56,276	0.058
johnson16-2-4	120	5,460	0.765	SC-NIP-r-t2	1,183	17,776	0.025

However, they have not used the automatic cut-generation option of the CPLEX solver in order to show the strength of the improved formulations. We also consider this classic formulation and present the corresponding solution times using the ILOG/CPLEX 12.7 package with the automatic cut generation on. This result is shown under the column IP_{BASE} in Table 2.2. In this table, we use "$>$" as a substitute of "$> 10,800$" to indicate that the process was terminated due to the upper time limit before concluding the search.

Given that the RTN and SC-NIP instances are very sparse, we added a preprocessing step to the algorithm and slightly modified the pruning process for these graphs. In the preprocessing step, vertices with degree of at most two are eliminated from the graph after recording the maximum weight of a clique that they contribute to. At the end, the result of the combinatorial B&B algorithm on the residual graph is compared with these values and the best solution is reported. Besides, at each node of the search tree, the pruning subroutine is called only if the edge density of the subgraph induced by the candidate list is at least 10%. This prevents the algorithm from performing the expensive spectral calculations on highly sparse subgraphs.

Computational results reveal the competitiveness of the CBQ algorithm. We could solve in-

Table 2.2: Solution time.

Name	W^*	CPU (sec.)			
		CBQ	Gouveia & Martins		IP_{BASE}
brock200-1	21,230	3,047.565	$>$		$>$
brock200-2	6,542	7.436	9,464.240	(F1)	$>$
brock200-3	10,303	55.905	$>$		$>$
brock200-4	13,967	188.031	$>$		$>$
C125.9	66,248	4,558.170	$>$		$>$
C250.9	-	$>$	$>$		$>$
c-fat200-1	7,734	0.483	3.870	(F61)	31.296
c-fat200-2	26,389	0.890	33.260	(F2)	49.671
c-fat200-5	168,200	$>$	155.300	(F1)	134.578
hamming6-2	32,736	4.437	0.300	(F11)	17.000
hamming6-4	396	0.031	1.970	(F1)	6.468
hamming8-2	-	$>$	$>$		$>$
hamming8-4	12,360	439.437	$>$		$>$
johnson16-2-4	3,808	84.687	$>$		$>$
johnson8-2-4	192	0.000	0.140	(F61)	0.421
johnson8-4-4	6,552	0.687	2.340	(F11)	65.171
keller4	6,745	42.218	$>$		$>$
MANN-a9	5,460	1.906	9.390	(F1)	130.344
p-hat300-1	3,321	3.281	1,273.050	(F2)	8,489.750
p-hat300-2	31,564	171.281	$>$		$>$
p-hat300-3	-	$>$	$>$		$>$
d1-RTN	4,524	7.280	14.680	(F62)	1,607.980
d3-RTN	5,859	68.874	1,565.550	(F62)	$>$
d7-RTN	7,424	193.047	799.360	(F5)	$>$
d15-RTN	7,424	389.672	$>$		$>$
SC-NIP-m-t1	343	1.374	5.770	(F62)	145.141
SC-NIP-r-t1	25,290	995.359	8.160	(F6)	5,134.550
SC-NIP-r-t2	15,188	32.312	0.700	(F6)	146.484

stances that all nine formulations of [25] failed to solve, and among those that were solved by both methods we could reach much better solution times with a few exceptions. While Gouveia and Martins could solve 16 out of 28 instances, the CBQ algorithm successfully found an optimal solution for 24 instances. The quality of CBQ algorithm can be particularly observed through its performance on the `brock` and `p-hat` families. In the `brock` family, the CBQ algorithm could solve all four instances very fast. `brock200-2` is the only instance in this family that could be solved in [25], but with a solution time of more than 2.5 hours. The CBQ algorithm solved this instance in less than 8 seconds. Among the instances of the `p-hat` family, only `p-hat300-1` could be solved in [25] with a solution time of more than 20 minutes; while the CBQ algorithm solved it in less than four seconds. Note that none of the formulations in [25] was the best uniformly over all solved instances. The only instance for which the CBQ algorithm failed to find an exact solution found by the best formulation of [25] was `c-fat200-5`.

The results of the CBQ algorithm presented in Table 2.2 use the solutions provided by the QCH heuristic as initial lower bounds. As it was mentioned previously, better heuristic results could be attained by implementing the QCH algorithm for each closed neighborhood (see Algorithm 3). Spending more time to get better initial bounds is usually preferable in large and dense graphs, where it can result in a considerable reduction in size of the search tree. Table 2.3 compares the quality of heuristic solutions obtained from QCH and QCH-N algorithms. In this table, $\tilde{W}$ denotes weight of the heuristic solution.

We repeated the experiments with the QCH-N lower bound. Although an optimal solution was reached in the heuristic phase for more instances, it did not lead to solving any new instance within the time limit of the B&B algorithm. Table 2.3 shows that the heuristic solution of QCH-N algorithm is optimal for 23 out of the 28 instances considered, while this number is only 16 for the QCH algorithm.

Finally, we present our results explicitly about quality of the quadratic relaxation (QR) bound in comparison with the sum of edge weights (SUM). In Table 2.4, #QR and #SUM denote the number of times that the algebraic bound and the combinatorial bound were used in Algorithm 7,

Table 2.3: Heuristic results.

		QCH		QCH-N	
Name	W^*	$\tilde{W}$	CPU (sec.)	$\tilde{W}$	CPU (sec.)
brock200-1	21,230	21,230	0.015	21,230	1.984
brock200-2	6,542	6,542	0.015	6,542	0.750
brock200-3	10,303	10,303	0.015	10,303	1.250
brock200-4	13,967	9,634	0.015	13,736	1.562
C125.9	66,248	53,145	0.000	65,416	0.734
C250.9	-	69,977	0.015	83,780	7.062
c-fat200-1	7,734	7,734	0.015	7,734	0.046
c-fat200-2	26,389	26,389	0.015	26,389	0.062
c-fat200-5	168,200	168,200	0.015	168,200	0.625
hamming6-2	32,736	32,736	0.000	32,736	0.046
hamming6-4	396	396	0.000	396	0.000
hamming8-2	-	800,624	0.046	800,624	9.796
hamming8-4	12,360	12,160	0.015	12,360	2.968
johnson16-2-4	3,808	3,608	0.000	3,808	0.328
johnson8-2-4	192	192	0.000	192	0.000
johnson8-4-4	6,552	6,552	0.000	6,552	0.046
keller4	6,745	6,745	0.015	6,745	0.796
MANN-a9	5,460	5,445	0.000	5,460	0.046
p-hat300-1	3,321	3,089	0.031	3,321	0.578
p-hat300-2	31,564	25,412	0.031	31,564	3.328
p-hat300-3	-	50,995	0.031	59,425	8.218
d1-RTN	4,524	4,524	2.609	4,524	0.265
d3-RTN	5,859	5,859	24.390	5,859	2.328
d7-RTN	7,424	7,244	64.000	7,424	6.687
d15-RTN	7,424	6,735	128.828	7,424	13.578
SC-NIP-m-t1	343	343	0.609	343	0.234
SC-NIP-r-t1	25,290	25,290	2.640	25,290	14.859
SC-NIP-r-t2	15,188	15,188	0.500	15,188	2.421

respectively. The numbers in parentheses represent the corresponding percentage. The last column, titled “avg. QR/SUM”, shows the average ratio of the algebraic bound (when calculated) to the sum of edge weights over all calls of the PRUNE function for each instance. This table includes only the instances that we could solve under the time constraint.

Table 2.4 shows that, on DIMACS instances, the QR bound was much tighter than SUM except for `c-fat200-1`, `c-fat200-2` and `hamming6-2`. Excluding these instances, the maximum average ratio is 0.76 for `MANN-a9`. The ratio is as good as 0.40 and 0.44 for `johnson16-2-4` and `keller4`, respectively. For the `brock, C, johnson, keller, MANN` and `p-hat` families, the number of times that the algorithm applied the combinatorial bound is negligible compared to the application of QR bound. Looseness of the QR bound on the `c-fat` family could explain failure of our algorithm in solving `c-fat200-5` under the time limit. On the RTN and SC-NIP, however, the QR bound did not perform as well as on the DIMACS graphs. While the number of times that the QR bound outperformed SUM is considerable—except for `SC-NIP-r-t1`—it was never the dominant pruning method and it performed very poorly in terms of the average ratio.

Table 2.4: Quality of the quadratic relaxation bound.

Name	#QR	#SUM	avg. QR/SUM
brock200-1	38,819,232 (99.82 %)	71,511 (0.18 %)	0.66
brock200-2	108,456 (99.82 %)	200 (0.18 %)	0.57
brock200-3	884,461 (99.82 %)	1,590 (0.18 %)	0.61
brock200-4	2,711,652 (99.85 %)	4,071 (0.15 %)	0.62
C125.9	26,539,316 (99.95 %)	13,835 (0.05 %)	0.72
c-fat200-1	5 (1.35 %)	366 (98.65 %)	1.27
c-fat200-2	0 (0.00 %)	4,861 (100.00 %)	1.30
hamming6-2	11,836 (36.68 %)	20,431 (63.32 %)	1.02
hamming6-4	790 (80.28 %)	194 (19.72 %)	0.70
hamming8-4	4,990,643 (97.13 %)	147,215 (2.87 %)	0.65
johnson16-2-4	3,076,874 (99.49 %)	15,649 (0.51 %)	0.40
johnson8-2-4	215 (89.96 %)	24 (10.04 %)	0.51
johnson8-4-4	10,863 (98.19 %)	200 (1.81 %)	0.55
keller4	704,846 (99.71 %)	2,061 (0.29 %)	0.44
MANN-a9	72,891 (90.18 %)	7,938 (9.82 %)	0.76
p-hat300-1	22,217 (98.43 %)	354 (1.57 %)	0.58
p-hat300-2	1,694,985 (99.13 %)	14,820 (0.87 %)	0.74
d1-RTN	574 (39.42 %)	882 (60.58 %)	3.87
d3-RTN	1,106 (32.22 %)	2,327 (67.78 %)	4.90
d7-RTN	1,449 (27.48 %)	3,824 (72.52 %)	5.33
d15-RTN	2,057 (21.74 %)	7,406 (78.26 %)	4.61
SC-NIP-m-t1	407 (35.42 %)	742 (64.58 %)	2.35
SC-NIP-r-t1	116 (0.04 %)	298,088 (99.96 %)	3.98
SC-NIP-r-t2	131 (12.31 %)	933 (87.69 %)	10.89

3. A LAGRANGIAN BOUND ON THE CLIQUE NUMBER AND AN EXACT ALGORITHM FOR THE MAXIMUM EDGE WEIGHT CLIQUE PROBLEM*

3.1 Introduction

Given a simple, undirected graph $G = (V, E)$, where $V = \{1, \dots, n\}$ is the set of vertices and E is the set of edges, a *clique* is a subset of vertices $C \subseteq V$ inducing a complete subgraph. A clique is called *maximal* if it is not a (proper) subset of a larger clique, and *maximum* if there is no larger clique in the graph. The cardinality of a maximum clique in G is called the *clique number* of the graph, and is denoted by $\omega(G)$. The *maximum clique (MC) problem*, which is to find a maximum clique in a graph, is one of the most popular problems of combinatorial optimization and a great deal of research has been dedicated to its study; see, e.g., the survey papers [55] and [23].

The *maximum edge weight clique (MEWC) problem* is a generalization of the MC problem to edge-weighted graphs. In an edge-weighted graph $G = (V, E, w^E)$, every edge $\{i, j\} \in E$ has a positive weight w_{ij}, and the edge weight of $C \subseteq V$ is defined as $W(C) = \sum_{\{i,j\} \in E(C)} w_{ij}$, where $E(C)$ denotes the set of edges with both endpoints in C. The MEWC problem is to find a clique with the maximum edge weight in G. If the weight of each edge is equal to 1, finding a maximum edge weight clique is equivalent to finding a maximum clique C^* with $W(C^*) = \binom{\omega(G)}{2}$. Therefore, the MEWC problem is at least as hard to solve as the MC problem, which is NP-hard [3]. The MC problem is also known to be hard to approximate to within $n^{1-\epsilon}$ for every $\epsilon > 0$ [56]. Similar to the MC problem, the MEWC problem finds applications in various fields, including computational biology [7, 8], computer graphics [4, 5], marketing [6], healthcare [9], and materials science [27].

In this chapter, we investigate some close connections between the MEWC and MC problems that lead to new results for both problems. First, we use a Lagrangian relaxation of an integer (linear) programming formulation of the MEWC problem to derive an analytic upper bound on the clique number of a graph. Then, upper bounds on the clique number are used in an upper-bounding

* This chapter is reprinted with permission from "A Lagrangian bound on the clique number and an exact algorithm for the maximum edge weight clique problem" by S. Hosseinian, D. B. M. M. Fontes, and S. Butenko [54]. Copyright © 2020, INFORMS.

scheme for the MEWC problem, which is employed in a combinatorial branch-and-bound (B&B) procedure for this problem. The performance of the proposed algorithm strongly depends on our ability to quickly compute nontrivial upper bounds on the clique number.

Several analytic (closed-form) upper bounds on the clique number of a graph have been proposed in the literature. The first bound of this type appeared in the work of Wilf [57], who showed that $\omega(G) \leq \rho(A_G) + 1$, where $\rho(A_G)$ denotes the spectral radius of the adjacency matrix A_G of G. Later, Amin and Hakimi [58] presented two bounds. They noted that $\omega(G) \leq \frac{3+\sqrt{9-8(n-m)}}{2}$ for connected graphs, with n and m denoting the number of vertices and edges, respectively. They also proved that $\omega(G) \leq N_{-1}(A_G) + 1$, where $N_{-1}(A_G)$ is the number of eigenvalues of the adjacency matrix of the graph not exceeding -1. Budinich [59] used complex analysis to show that $\omega(G) \leq n - \frac{\mathit{rank}(A_{\bar{G}})}{2}$, where $A_{\bar{G}}$ denotes the adjacency matrix of the complement graph $\bar{G}$ of G, and $\mathit{rank}(A_{\bar{G}})$ is the rank of this matrix. Among these bounds, only the first one of Amin and Hakimi [58] is computable in linear time. The others involve spectral calculations on the adjacency matrix of the graph (or its complement) and require $\mathcal{O}(n^3)$ arithmetic operations. The upper bound derived in this chapter provides another linear-time computable alternative to the first Amin-Hakimi bound. It should be noted that our bound was obtained as a result of an attempt of obtaining a tight Lagrangian relaxation-based upper bound for the MEWC problem. Consequently, coloring-based bounds proved to be superior within the proposed B&B framework for the MEWC problem; however, the obtained analytic upper bound on the clique number is still of interest as a nontrivial, closed-form, and easily computable bound for the classical MC problem.

A closely related problem to the MEWC problem is the *maximum diversity problem*, which is to find a maximum edge weight clique of cardinality not exceeding a given bound k in a complete edge-weighted input graph. As suggested by some authors [60, 61], an instance of the MEWC problem can be transformed to a complete graph by adding dummy edges with sufficiently-large negative weights and then solved as an instance of the maximum diversity problem with $k = |V|$. In this regard, these two names have been used interchangeably in the literature to refer to the maximum diversity problem. The exact solution methods proposed for the maximum diversity problem include combinatorial B&B [62] and branch-and-cut algorithms based on integer (linear) programming formulations [10, 11, 12, 14, 13]. Several heuristic and metaheuristic methods have also been applied to this problem, including tabu search [15, 16, 17], memetic search [18], scatter search [19], and greedy randomized adaptive search procedure [20, 21].

To the best of our knowledge, the only works that focused on the MEWC problem, i.e., with non-complete input graphs, at the time of completion of the present work, were [24, 25, 26, 2, 63]. More specifically, Pullan [24] extended the phased local search method proposed for the MC problem in [64] to address the MEWC problem. Gouveia and Martins [25] studied existing integer programming formulations of the maximum diversity problem to adapt them for sparse graphs and introduced a set of new formulations for the MEWC problem. Recently, Hosseinian et al. [26] developed a construction heuristic algorithm based on solving a quadratically constrained quadratic problem. In [2], the same authors introduced a quadratic programming formulation for the MEWC problem, and showed the correspondence between local optima of the continuous problem and special structures in the underlying graph. They also presented an exact combinatorial B&B algorithm that uses a relaxation of the new formulation to prune the search tree. In addition, Fontes et al. [63] proposed a biased random-key genetic algorithm for this problem. We refer to [27] for a detailed review of these methods.

3.2 A Lagrangian Relaxation Bound on the Clique Number

A well-known integer (linear) programming formulation of the MEWC problem on a graph $G = (V, E, w^E)$ with positive edge weights is as follows [11]:

$$\begin{aligned}
W^* = \max \quad & \sum_{\{i,j\}\in E} w_{ij}\, y_{ij} \\
\text{s.t.} \quad & y_{ij} \le x_i \;\text{ and }\; y_{ij} \le x_j, && \forall \{i,j\} \in E, \\
& x_i + x_j \le 1, && \forall \{i,j\} \notin E, \\
& x_i \in \{0,1\}, && \forall i \in V, \\
& y_{ij} \in \{0,1\}, && \forall \{i,j\} \in E.
\end{aligned} \tag{3.1}$$

In this formulation, every vertex $i \in V$ is represented by a variable x_i and every edge $\{i, j\} \in E$ by a variable y_{ij}. Every feasible solution of (3.1) induces a clique $C = \{i \in V \mid x_i = 1\}$ with $E(C) = \{\{i, j\} \in E \,|\, y_{ij} = 1\}$. Therefore, an optimal solution of (3.1) characterizes a clique with the maximum total sum of edge weights in the induced subgraph.

Consider the full Lagrangian relaxation of (3.1), except for integrality of the variables, i.e.,

$$\begin{aligned}\mathcal{Z}(\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \boldsymbol{\mu}^3) = \max_{\mathbf{x},\mathbf{y}} \quad & \sum_{\{i,j\}\in E} w_{ij}y_{ij} + \sum_{\{i,j\}\in E} \mu^1_{ij}(x_i - y_{ij}) + \\ & \sum_{\{i,j\}\in E} \mu^2_{ij}(x_j - y_{ij}) + \sum_{\{i,j\}\notin E} \mu^3_{ij}(1 - x_i - x_j) \\ \text{s.t.} \quad & x_i \in \{0,1\}, \qquad \forall i \in V, \\ & y_{ij} \in \{0,1\}, \qquad \forall \{i,j\} \in E,\end{aligned} \tag{3.2}$$

where $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$, $\boldsymbol{\mu}^3 \geq \mathbf{0}$ denote the vectors of Lagrangian multipliers (dual variables) corresponding to the first, second and third set of constraints in (3.1), respectively. Restricting the dual variables in (3.2) to the line $\mu^1_{ij} = \mu^2_{ij} = \mu^3_{ij} = \mu$, $\forall i, j \in V$, we obtain a simpler problem:

$$\begin{aligned}z(\mu) = \max_{\mathbf{x},\mathbf{y}} \quad & \sum_{\{i,j\}\in E} (w_{ij}y_{ij} + \mu(x_i + x_j - 2y_{ij})) + \sum_{\{i,j\}\notin E} \mu(1 - x_i - x_j) \\ \text{s.t.} \quad & x_i \in \{0,1\}, \qquad \forall i \in V, \\ & y_{ij} \in \{0,1\}, \qquad \forall \{i,j\} \in E.\end{aligned} \tag{3.3}$$

Let $\mathcal{L}(\mathbf{x}, \mathbf{y}, \mu)$ denote the objective function of (3.3). Then,

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{y}, \mu) &= \sum_{\{i,j\}\in E} (w_{ij}y_{ij} + \mu(x_i + x_j - 2y_{ij})) + \sum_{\{i,j\}\notin E} \mu(1 - x_i - x_j) \\ &= \sum_{\{i,j\}\in E} y_{ij}(w_{ij} - 2\mu) + \mu \left(\sum_{\{i,j\}\in E} (x_i + x_j) + \sum_{\{i,j\}\notin E} (1 - x_i - x_j) \right) \\ &= \sum_{\{i,j\}\in E} y_{ij}(w_{ij} - 2\mu) + \mu \left(\bar{m} + \sum_{i\in V} x_i(2d_i - n + 1) \right),\end{aligned} \tag{3.4}$$

where d_i denotes the degree of a vertex $i \in V$, n is the number of vertices, and $\bar{m} = \binom{n}{2} - |E|$ represents the number of edges in the complement graph of G. Equation (3.4) implies that the restricted Lagrangian relaxation problem (3.3) is separable with respect to variables corresponding to vertices and edges of G, so it can be rewritten as follows:

$$z(\mu) = \max_{\mathbf{y} \in \{0,1\}^{|E|}} \sum_{e \in E} y_e(w_e - 2\mu) + \mu \left(\bar{m} + \max_{\mathbf{x} \in \{0,1\}^n} \sum_{v \in V} x_v(2d_v - n + 1) \right). \tag{3.5}$$

Optimal solutions of both optimization problems in (3.5) are determined by the signs of the coefficients of the binary decision variables. Therefore, $z(\mu)$ can be calculated according to (3.6) for every $\mu \geq 0$:

$$z(\mu) = \sum_{e \in E^+} (w_e - 2\mu) + \mu \left(\bar{m} + \sum_{v \in V^+} (2d_v - n + 1) \right), \tag{3.6}$$

where $V^+ \subseteq V$ and $E^+ \subseteq E$ are defined as follows:

$$V^+ = \{v \in V \mid d_v \geq (n-1)/2\}, \tag{3.7}$$

$$E^+ = \{e \in E \mid w_e \geq 2\mu\}. \tag{3.8}$$

Note that V^+ only depends on the structure of G, while E^+ is a function of weights and the Lagrangian multiplier μ. Minimizing $z(\mu)$ over the non-negative values of μ gives an upper bound on the optimal value of the MEWC problem. That is,

$$W^* \leq \min_{\boldsymbol{\mu} \geq \mathbf{0}} \mathcal{Z}(\boldsymbol{\mu}) \leq \min_{\mu \geq 0} \sum_{e \in E^+} (w_e - 2\mu) + \mu \left(\bar{m} + \sum_{v \in V^+} (2d_v - n + 1) \right), \tag{3.9}$$

where $\boldsymbol{\mu} = (\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \boldsymbol{\mu}^3)$. In (3.9), the first inequality is the weak duality and the second one is due to the fact that $z(\mu)$ is a restriction of $\mathcal{Z}(\boldsymbol{\mu})$. We use this result to get an upper bound on the clique number of the graph. The following proposition serves this purpose.

Proposition 6. *Given an undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, let $b = \binom{n}{2} - m + \sum_{v \in V^+}(2d_v - n + 1)$, where d_v denotes degree of a vertex $v \in V$ and V^+ is defined* *in* (3.7)*. Then,* $a = \left\lfloor \frac{1+\sqrt{4b+1}}{2} \right\rfloor$ *gives an upper bound on* $\omega(G)$*.*

Proof. Consider an edge-weighted version of G with $w_{ij} = 1,\ \forall\{i,j\} \in E$. Then, by (3.9),

$$\bar{W} = \min_{\mu \geq 0} \sum_{e \in E^+} (1 - 2\mu) + \mu b \tag{3.10}$$

is an upper bound on the optimal value of the MEWC problem on G, which equals $\binom{\omega(G)}{2}$. Note that the parameter b depends only on the structure of the graph and is invariant of the edge weights. Given the definition of E^+, the objective function of (3.10) is increasing for $\mu \geq \frac{1}{2}$, so we can restrict the optimization interval to $[0, \frac{1}{2}]$. Since $E^+ = E$ for all values of $\mu \in [0, \frac{1}{2}]$, the objective function of (3.10) is linear and reaches its minimum at $\mu = 0$ or $\mu = \frac{1}{2}$. Therefore, $\binom{\omega(G)}{2} \leq \min\{|E|\,,\, \frac{b}{2}\} \leq \frac{b}{2}$, which implies that $\omega(G) \leq \left\lfloor \frac{1+\sqrt{4b+1}}{2} \right\rfloor$. □

This bound equals $\omega(G)$ if the graph can be partitioned into two cliques that are either very densely or very sparsely connected to each other. Two extreme cases are complete graphs and graphs whose vertex sets are independent union of two cliques, on which the proposed bound can be easily checked to be sharp. Clearly, the bound is loose on sparse graphs due to the term $\bar{m} = \binom{n}{2} - m$.

We illustrate Proposition 6 on the graph of Figure 3.1. This graph has $n = 7$ vertices and $m = 13$ edges, and its clique number is $\omega(G) = 3$.

$$b = \binom{n}{2} - m + \sum_{v \in V^+} (2d_v - n + 1) = 21 - 13 + 5(8 - 7 + 1) = 18 \tag{3.11}$$

$$a = \left\lfloor \frac{1+\sqrt{4b+1}}{2} \right\rfloor = \left\lfloor \frac{1+\sqrt{73}}{2} \right\rfloor = \lfloor 4.77 \rfloor = 4 \geq \omega(G) \tag{3.12}$$

Note that this graph has 5 vertices of degree 4, so absence of a clique of cardinality 5 is not evident without exploring adjacency of the vertices. In particular, using the first bound by Amin and Hakimi [58], we obtain $\omega(G) \leq \left\lfloor \frac{3+\sqrt{9-8(7-13)}}{2} \right\rfloor = 5$. None of the spectral bounds, presented in Section 3.1, will generate an upper bound better than 4 for this graph.

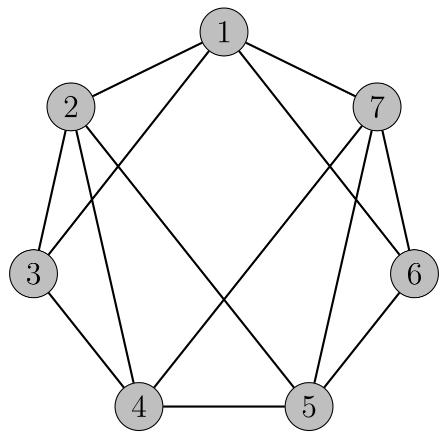

Figure 3.1: Example graph for the analytic bound on the clique number.

3.3 An Exact Solution Method for the MEWC Problem

Consider the univariate Lagrangian relaxation problem (3.9) presented in the previous section. The objective function of this minimization problem is piecewise linear and convex, thus it can be solved efficiently to get an upper bound on W^*. Such an upper bound, however, is usually too loose to be used in a B&B algorithm. In fact, it can be shown that the optimal value of the univariate Lagrangian relaxation problem is no better than sum of the $\frac{b}{2}$ largest edge weights in the graph, where b is given in Proposition 6. A formal proof for this statement is presented in Appendix A. This result implies that a similar approach may be taken with any upper bound $\bar{\omega} \geq \omega(G)$ better than a (in Proposition 6) to tighten the bound on W^*. Evidently, given availability of such $\bar{\omega}$, this approach is disadvantageous as it will not generate an upper bound better than sum of the $\binom{\bar{\omega}}{2}$ largest edge weights in the graph. Motivated by this observation, in Section 3.3.1, we present an upper-bounding method for the MEWC problem that takes advantage of known upper bounds on the clique number and exploits adjacency of edges in the graph to draw a tighter bound on W^*.

We use this upper-bounding method in the pruning subroutine of a combinatorial B&B procedure. The combinatorial B&B procedure is an implicit enumeration of cliques in the graph, which has been vastly used before in the algorithms proposed for the MC problem with different pruning methods and implementation details; see for example [47, 48, 49, 50, 51, 52]. The procedure starts with a trivial clique, i.e., a single vertex, and continues by examining all cliques in the graph until a clique with the maximum weight is found. The systematic search is done through a tree traversal. Every node of the search tree corresponds to a subgraph induced by the union of a clique C and a list of vertices, called *candidate list*. The candidate list is composed of the vertices in $V \backslash C$ that are adjacent to all vertices in C, so appending any of them to C would result in a larger, and hence heavier, clique. A node is pruned, i.e., the corresponding subgraph is excluded from further investigation, if an upper bound on the maximum clique-weight in the corresponding subgraph is not greater than a known solution. If a node is not pruned, it branches by expanding the incumbent clique via appending a vertex from the candidate list to it and so generating a new node of the tree. In Section 3.3.2, we present details of the algorithm, including the pruning subroutine for the MEWC problem.

3.3.1 Upper-bounding method

Suppose $\bar{\omega} \geq \omega(G)$, an upper bound on the clique number of an edge-weighted graph $G = (V, E, w^E)$, is known. Then, sum of the $\binom{\bar{\omega}}{2}$ largest edge weights of G gives a trivial upper bound for the MEWC problem. This approach is disadvantageous as it ignores the structure of the graph beyond the knowledge of $\bar{\omega}$. To obtain a better bound, we need to take into account the adjacency of edges.

Suppose C^* is a maximum edge weight clique of G. For every vertex $i \in V$, let

$$\Gamma(i) = \begin{cases} \frac{1}{2} \sum_{j \in C^* \backslash \{i\}} w_{ij}, & i \in C^* \\ 0, & i \in V \backslash C^*. \end{cases} \tag{3.13}$$

Then, $W^* = W(C^*)$ can be expressed in terms of attributes of the vertices rather than edges, i.e.,

$$W(C^*) = \sum_{i \in C^*} \Gamma(i). \tag{3.14}$$

Since $|C^*| \leq \omega \leq \bar{\omega}$, the summations in (3.13) and (3.14) are over at most $\bar{\omega} - 1$ and $\bar{\omega}$ vertices, respectively. Therefore, $\Gamma(i)$ and $W(C^*)$ are bounded from above according to (3.15) and (3.16):

$$\Gamma(i) \leq \bar{\Gamma}(i) = \frac{1}{2}\sum_{k=1}^{\bar{\omega}-1} w_e^{(k)} : e \in \delta_i, \ \forall i \in V, \tag{3.15}$$

$$W(C^*) \leq \sum_{k=1}^{\bar{\omega}} \bar{\Gamma}^{(k)}, \tag{3.16}$$

where δ_i is the set of edges incident to a vertex $i \in V$ and the superscript (k) denotes the k-th largest value in the corresponding set. Note that the computational complexity of calculating this upper bound on $W(C^*)$ is $\mathcal{O}(m \log n)$, which is no worse than sorting the edges according to their weights.

We refine (3.15)-(3.16) further based on the special structure of the subgraphs visited throughout the search procedure. Recall that each subgraph is induced by the union of an incumbent clique C and its candidate list L, denoted by $G[C \cup L]$. Consider G', the minor of $G[C \cup L]$ generated by merging all vertices of C into a single vertex v. In this graph, the weights of the edges connecting v to vertices of L are defined as follows:

$$w'_{iv} = \sum_{j \in C} w_{ij}, \ \forall i \in L. \tag{3.17}$$

Clearly, the difference between the maximum edge weights of cliques in $G[C \cup L]$ and G' is equal to $W(C)$. Hence, we may look for an upper bound on the optimal value of the MEWC problem on G', which we denote by $W^*_{G'}$. Since v is adjacent to all vertices of L in G', we can restate (3.15) as follows:

$$\Gamma'(i) \leq \bar{\Gamma}'(i) = w'_{iv} + \frac{1}{2}\sum_{k=1}^{\bar{\omega}_L-1} w_e^{(k)} : e \in \delta_i^L, \ \forall i \in L, \tag{3.18}$$

where $\Gamma'(i)$ is the contribution of a vertex $i \in L$ to $W^*_{G'}$, $\bar{\omega}_L$ is an upper bound on the clique number of $G'[L] = G[L]$, and δ_i^L denotes the set of edges incident on a vertex $i \in L$ in $G[L]$. Therefore,

$$W^*_{G'} \leq \sum_{k=1}^{\bar{\omega}_L} \bar{\Gamma}'^{(k)}. \tag{3.19}$$

This leads to the following bound which is employed in our algorithm:

$$W^*_{G[C\cup L]} \leq W(C) + \sum_{k=1}^{\bar{\omega}_L} \bar{\Gamma}'^{(k)}. \tag{3.20}$$

In order to calculate $\bar{\omega}_L$ in our algorithm, we consider two *chromatic* methods from the literature that underlie recent advancements in exact algorithms of the MC problem: the method proposed by Tomita and Kameda [49] and the one by San Segundo et al. [52]. The computational effort required by these methods is comparable to that of analytic bounds, while results of experiments with benchmark instances have shown that they usually generate much tighter upper bounds on the clique number. High quality upper bounds, such as *Lovász number* [65] and *η-Bound* [66], on the other hand, require solving computationally expensive optimization problems and are not suited for B&B methods.

The first chromatic method that we consider is a sequential vertex-coloring heuristic based on a preordered list of vertices. It is well known that any proper coloring of vertices, i.e., assignment of colors such that every two adjacent vertices have different colors, generates an upper bound on the clique number of a graph. Although such upper bounds could be arbitrarily loose for special graphs [67], several algorithms for the MC problem have successfully used this method [48, 68, 49, 50, 51]. The second method is based on a MaxSAT encoding of the MC problem that was originally proposed by Li and Quan [69]. We use the implementation of this method presented in [52] and refer to it as a chromatic method, as it was presented in the context of graph coloring without the overhead of an explicit MaxSAT encoding. It has been shown in [52] that this bound could be even better than the chromatic number, i.e., the minimum number of colors in a proper coloring, of the graph. For the MC problem, the algorithm that employs the second method generally outperforms the ones that use the first method and its improvements, e.g., [50] and [51]. However, this is not the case for the MEWC problem, as our experimental results show, because of different pruning criteria used for these two problems and the overhead of calculating *pruning threshold* in the latter method.

3.3.2 Algorithm

The algorithm starts with building two arrays of vertices, namely U and L, in the initialization step. The vertices are sorted according to an initial ordering and stored in array U. This ordering aims to keep the subproblems formed during the B&B process small. The array L is a copy of U, in which the vertices are colored based on their positions in the array. These two arrays are then used by the BRANCH function, which is the main procedure of the combinatorial B&B algorithm. All input arguments of this function are global variables and are updated as the search proceeds. Algorithm 8 outlines the main procedure.

Algorithm 8 Main method

```
1: function SOLVE_MEWC($G$)
2:     $C = \emptyset$ ;  $W = 0$
3:     $C^* = \emptyset$ ;  $W^* = 0$
4:     $(U, L)$ = INITIALIZE($G$)
5:     BRANCH($G, U, L, C, C^*, W, W^*$)
6:     return $(C^*, W^*)$
7: end function
```

The INITIALIZE function (see Algorithm 12 in Appendix B) stores the vertices in array U such that the k-th vertex in U, denoted by $U[k]$, has the smallest degree in the subgraph induced by $\{U[1], U[2], \ldots, U[k]\}$. If there are two (or more) vertices with minimum degree in that subgraph, their relative order in U is determined based on degrees of their neighbors. Let G^k denote the subgraph induced by the first k vertices in U. If more than one vertex in G^k have the minimum degree, then $U[k]$ will be the vertex with the minimum sum of neighbors' degrees, denoted by σ in Algorithm 12. If there is still a tie among some vertices, $U[k]$ is chosen arbitrarily. This method of initialization was presented as a part of the MCR algorithm [49] and has been used in almost all succeeding combinatorial B&B algorithms for the MC problem, but its main idea was first proposed by Carraghan and Pardalos [47]. Initial coloring of the vertices follows the fact that the number of colors required to properly color a graph is no more than the maximum degree of the vertices plus one. Therefore, the number of colors to properly color G^k will be no more than $\min\{k, \Delta(G) + 1\}$, where $\Delta(G)$ denotes the maximum degree of a vertex in G. The INITIALIZE function gives distinct colors to the first $\Delta(G) + 1$ vertices in the sorted array and repeats the last color for all the remaining vertices. Note that the initial coloring is not necessarily proper, but the color (natural number) assigned to the vertex $L[k]$ for every $k \in \{1, \ldots, n\}$ is indeed an upper bound on the clique number of G^k.

The tree search is done by the BRANCH function in a recursive manner. In every call, this function inputs an incumbent clique C and its weight W, the best-known solution C^* and its weight W^*, and two lists of candidate vertices to expand the incumbent clique, denoted by U^p and L^p. Initially, C and C^* are empty, W and W^* are zero, and the candidate lists, i.e., U and L, involve all vertices of the graph. Similar to U and L, the candidate lists at each node of the search tree, i.e., subproblems of the BRANCH function, are comprised of the same set of vertices. Vertices in U^p are uncolored and keep the relative order of vertices in U. Vertices in L^p are colored and reordered based on the subproblem coloring. Actually, the recursion can be implemented using a single array of candidate vertices, but experimental results on the maximum clique problem have shown that keeping the initial ordering would generate, on average, tighter bounds on the clique number [50, 70]. Implementation details of this function are presented in Algorithm 9.

The BRANCH function operates as follows: first, it calls the PRUNE function to determine if the corresponding subgraph should be processed further. If so, the last vertex in array L^p is added to C, resulting in a larger clique. Then, a new subproblem is formed based on this new clique. The uncolored candidate list of the new subproblem, denoted by U_v, is comprised of all vertices in the parent candidate list U^p that are connected to the newly-added vertex. U_v being empty implies that the current clique is maximal, thus its weight is compared against W^* and the better solution is kept. Otherwise, the BRANCH function calls itself on the new subproblem. Prior to this, the colored candidate list for the new subproblem, denoted by L_v, is generated through the SUBCOLOR function. The two chromatic methods that we consider require different implementations of the SUBCOLOR function. That is why we used a superscript $\bullet$ for this function in Algorithm 9 (line 13)

Algorithm 9 Combinatorial B&B procedure

```
1: function BRANCH($G, U^p, L^p, C, C^*, W, W^*$)
2:     while $U^p \neq \emptyset$ do
3:         $q = $ PRUNE($G, U^p, L^p, C, W, W^*$)
4:         if $q = $ true then                                    ▷ the node is pruned
5:             return
6:         else                                                  ▷ the node is not pruned
7:             $v \leftarrow$ last vertex in $L^p$
8:             $W_v = \sum_{u \in C} w_{uv}$
9:             $C \leftarrow C \cup \{v\}$
10:            $W \leftarrow W + W_v$
11:            $U_v = $ an array of neighbors of $v$ in $U^p$ with the same relative order
12:            if $U_v \neq \emptyset$ then
13:                $L_v = $ SUBCOLOR$^\bullet(\ldots)$
14:                BRANCH($G, U_v, L_v, C, C^*, W, W^*$)
15:            else if $W > W^*$ then
16:                $C^* \leftarrow C$ ; $W^* \leftarrow W$
17:            end if
18:            $C \leftarrow C \backslash \{v\}$
19:            $W \leftarrow W - W_v$
20:        end if
21:        $L^p \leftarrow L^p \backslash \{v\}$
22:        $U^p \leftarrow U^p \backslash \{v\}$
23:    end while
24:    return
25: end function
```

to indicate a template function. We represent the functions corresponding to the first and second chromatic methods by superscripts 1 and 2, respectively.

The function SUBCOLOR^1 (see Algorithm 13 in Appendix B) implements the heuristic graph coloring method proposed by Tomita and Kameda [49]. This function inputs U_v and partitions it into the union of K independent sets. A proper coloring of $G[U_v]$ is attained by giving a distinct color to each independent set I_k for every $k \in \{1, \ldots, K\}$. The vertices are then stored in array L_v in a non-decreasing order of their colors. As a result, the color of the i-th vertex in L_v is an upper bound on the clique number of the subgraph induced by the first i vertices in U_v. In particular, the color of the last vertex in L_v is an upper bound on the clique number of $G[U_v]$.

An improvement of this method was later presented in [50] via introducing a so-called *re-*

coloring procedure. More recently, San Segundo et al. [52] proposed a novel recoloring method based on the MaxSAT encoding of the MC problem. We will also use this method to draw an upper bound on the clique number, and provide its details via SUBCOLOR^2 function. But, we first present the pruning subroutine of our algorithm based on the upper-bounding method presented in Section 3.3.1.

In addition to computing the bounds on the clique number using SUBCOLOR^1 and SUBCOLOR^2 procedures, we also considered the sequential elimination method proposed by Gendron et al. [71]. However, the results of numerical experiments have shown that applying this method to enhance the bound on the clique number increases the CPU time spent on execution of the proposed B&B algorithm, and hence, are not reported here. The inferior performance can be explained by the fact that applying the sequential elimination algorithm increases the running time taken to compute the upper bound on the clique number by the factor of $\mathcal{O}(n^2)$ for a graph on n vertices.

Algorithm 10 demonstrates the pruning process. The PRUNE function operates as follows: the color of the last vertex in L^p gives an upper bound $\bar{\omega}$ on the clique number of the subgraph induced by the candidate list U^p. For every $i \in U^p$, $\bar{\Gamma}'(i)$ is calculated according to (3.18) and is stored in array Γ (lines 4-21 of Algorithm 10). This array is then sorted in a non-increasing order of elements, and the sum of its first $\bar{\omega}$ components is taken as W'. The corresponding subgraph is excluded from further investigation if $W + W'$ is not strictly larger than the weight of the best-known solution, i.e., W^*.

We conclude this section with details of the second chromatic method. A major parameter in implementation of the method proposed in [52] is the *pruning threshold*, denoted by T in Algorithm 11. The pruning threshold is the maximum number of colors that can be used in a proper coloring of vertices such that the corresponding node is pruned. This method aims to exclude some vertices from the vertex-coloring process when the number of colors exceeds the pruning threshold. In an ideal case, the number of colors remains at the threshold and the node is pruned. Algorithm 11 demonstrates details of calculating the pruning threshold for the MEWC problem in our algorithm.

Algorithm 10 Pruning procedure

```
1: function PRUNE($G, U^p, L^p, C, W, W^*$)
2:     $q =$ false;  $W' = 0$
3:     $\bar{\omega} =$ color of the last vertex in $L^p$
4:     $\Gamma$ = an array of size $|U^p|$ with all elements equal to zero
5:     for $i = 1$ to $|U^p|$ do
6:         $v \leftarrow U^p[i]$
7:         for $j = 1$ to $|C|$ do
8:             $u \leftarrow C[j]$;  $\Gamma[i] \leftarrow \Gamma[i] + w_{uv}$
9:         end for
10:        $\delta^i$ = an empty array of size $|U^p|$
11:        for $j = 1$ to $|U^p|$ do
12:            $u \leftarrow U^p[j]$
13:            if $\{u, v\} \in E$ then $\delta^i[j] = \frac{1}{2} w_{uv}$
14:            else $\delta^i[j] = 0$
15:            end if
16:        end for
17:        sort $\delta^i$ in a non-increasing order of elements
18:        for $k = 1$ to $\bar{\omega} - 1$ do
19:            $\Gamma[i] \leftarrow \Gamma[i] + \delta^i[k]$
20:        end for
21:    end for
22:    sort $\Gamma$ in a non-increasing order of elements
23:    for $k = 1$ to $\bar{\omega}$ do
24:        $W' \leftarrow W' + \Gamma[k]$
25:    end for
26:    if $W' \leq W^* - W$ then $q =$ true
27:    end if
28:    return $q$
29: end function
```

Let $W'_t = \sum_{k=1}^{t} \bar{\Gamma}'^{(k)}$, then the threshold of our pruning method is given by $T = \text{argmax}\{t | W'_t \leq W^* - W\}$, where W^* is the weight of the best-known solution and W is the weight of the incumbent clique. As shown by Algorithm 11, computing T involves several iterations of a time-consuming part of the pruning process, i.e., updating and sorting contributions of vertices. Note that we are not replacing this part of Algorithm 10 with a simple comparison between the pruning threshold and the color of the last vertex in the candidate list. In fact, the PRUNE function may be called several times at every node of the search tree. The best-known solution may be updated in between

Algorithm 11 Pruning threshold for the MEWC problem

```
function THRESHOLD($G, U_v, C, W, W^*$)
    $\Gamma$ = an array of size $|U_v|$ with all elements equal to zero
    for $i = 1$ to $|U_v|$ do
        initialize $\Gamma[i]$ according to lines 7-9 of Algorithm 10
        form array $\delta^i$ according to lines 10-17 of Algorithm 10
    end for
    $W_t = \max\{\Gamma[i] \; : \; 1 \leq i \leq |U_v|\}$
    if $W_t > W^* - W$ then $T \leftarrow 0$
    else
        $t = 0$ ; $W_t = 0$
        while $W_t \leq W^* - W$ do
            $t \leftarrow t + 1$
            if $t = |U_v|$ then go to line 28
            else
                for $i = 1$ to $|U_v|$ do
                    $\Gamma[i] \leftarrow \Gamma[i] + \delta^i[t]$
                end for
                $\tilde{\Gamma} \leftarrow \Gamma$
                sort $\tilde{\Gamma}$ in a non-increasing order of elements
                $W_t \leftarrow 0$
                for $k = 1$ to $t + 1$ do
                    $W_t \leftarrow W_t + \tilde{\Gamma}[k]$
                end for
            end if
        end while
        $T \leftarrow t$
    end if
    return $T$
end function
```

these calls, which can result in pruning the subsequent children of the node.

Given a pruning threshold T, the SUBCOLOR^2 function (see Algorithm 14 in Appendix B) operates as follows: consider a proper coloring of the graph, i.e., a partition of vertices into independent sets. A set of k colors is called *inconsistent* if there is no clique of cardinality k in the graph with one vertex in each set. If the number of colors is greater than the clique number, then there exists a color that forms an inconsistent set with some others. The procedure starts by assigning at most T colors to vertices such that each color identifies an independent set. If some vertices are still left, every time that a color $s \geq T+1$ is assigned to an uncolored vertex v, the algorithm looks for two colors $k_1, k_2 \leq T$ that form an inconsistent set with s (see Algorithm 15 in Appendix B). If such a set is found, k_1 and k_2 are marked as *forbidden* and are not considered to find more inconsistent sets. More importantly, vertex v is excluded from the assigned color set. After all vertices are processed, the colored ones are placed in array L^p in a non-decreasing order of colors. Note that under this coloring method, the arrays U^p and L^p do not necessarily contain the same set of vertices.

3.4 Computational Experiments

In this section, we present our computational results on 41 benchmark instances taken from the DIMACS dataset [53]. DIMACS instances are originally unweighted; Pullan [24] proposed an edge-weighted version of these graphs, named DIMACS-EW, which has been used in later works on this problem. The weight of an edge $\{i, j\} \in E$ in DIMACS-EW graphs is given by $w_{ij} = (i + j) \bmod 200 + 1$. Table 3.1 presents characteristics of the test instances. In this table, $|V|$, $|E|$ and "Density" denote number of vertices, number of edges, and edge density of the graph, respectively.

All algorithms proposed in this chapter were implemented in C++, compiled using Visual C++ and executed on an Intel® Core i-7 CPU @2.90 GHz, 8 GB RAM computer with Windows 8.1 OS. The spectral upper bounds on the clique number were computed using Intel® Math Kernel Library on the same system.

Table 3.2 provides results concerning the quality of the Lagrangian relaxation bound, i.e., a in Proposition 6, along with other analytic bounds proposed in the literature, as well as the coloring-based bound of [49]. In addition, Table 3.3 reports the corresponding CPU times (in milliseconds). As it was mentioned earlier, three of these bounds require spectral calculations on the adjacency matrix of the graph (or its complement). Given the difference in the required computational effort, we distinguish them from the first bound of Amin and Hakimi [58] and the Lagrangian bound proposed here. In Tables 3.2 and 3.3, "Wilf", "A&H", and "Budinich" refer to the bounds proposed in [57], [58], and [59], respectively. The column "LRB" shows the results for the Lagrangian

Table 3.1: Characteristics of the test instances (combinatorial algorithm).

Name	$\|V\|$	$\|E\|$	Density	Name	$\|V\|$	$\|E\|$	Density
brock200-1	200	14,834	0.75	johnson8-2-4	28	210	0.56
brock200-2	200	9,876	0.50	johnson8-4-4	70	1,855	0.77
brock200-3	200	12,048	0.61	keller4	171	9,435	0.65
brock200-4	200	13,089	0.66	MANN-a9	45	918	0.93
C125.9	125	6,963	0.90	p-hat300-1	300	10,933	0.24
C250.9	250	27,984	0.90	p-hat300-2	300	21,928	0.49
c-fat200-1	200	1,534	0.08	p-hat300-3	300	33,390	0.74
c-fat200-2	200	3,235	0.16	p-hat500-1	500	31,569	0.25
c-fat200-5	200	8,473	0.43	p-hat500-2	500	62,946	0.50
c-fat500-1	500	4,459	0.04	p-hat700-1	700	60,999	0.25
c-fat500-10	500	46,627	0.37	p-hat700-2	700	121,728	0.50
c-fat500-2	500	9,139	0.07	san200-0.7-1	200	13,930	0.70
c-fat500-5	500	23,191	0.19	san200-0.7-2	200	13,930	0.70
DSJC500-5	500	62,624	0.50	san200-0.9-1	200	17,910	0.90
gen200-p0.9-44	200	17,910	0.90	san200-0.9-2	200	17,910	0.90
gen200-p0.9-55	200	17,910	0.90	san200-0.9-3	200	17,910	0.90
hamming6-2	64	1,824	0.91	san400-0.5-1	400	39,900	0.50
hamming6-4	64	704	0.35	sanr200-0.7	200	13,868	0.70
hamming8-2	256	31,616	0.97	sanr200-0.9	200	17,863	0.90
hamming8-4	256	20,864	0.64	sanr400-0.5	400	39,984	0.50
johnson16-2-4	120	5,460	0.77				

relaxation bound presented in Section 3.2. Finally, the column "T&K" reports results for the coloring-based bounds according to Tomita and Kameda [49].

It is interesting to note that, with the exception of `c-fat` and `p-hat` families that mostly involve sparse graphs, the Lagrangian relaxation bound is relatively close to the spectral bound proposed by Wilf [57], i.e., $\rho(A_G) + 1$. Aside from these two families, the Lagrangian relaxation bound is better than the first bound of Amin and Hakimi [58] on all the other graphs, with the same time complexity.

Next, we compare the performance of the presented algorithm for the MEWC problem with the results reported by Gouveia and Martins [25], and Hosseinian et al. [2], as well as EW-CLIQUE algorithm by Shimizu et al. [72]. The first two works have considered instances with no more than 300 vertices from `brock`, `C`, `c-fat`, `hamming`, `johnson`, `keller`,

Table 3.2: Analytic and coloring-based bounds on the clique number.

		Spectral			Linear		Coloring
Name	$\omega(G)$	Wilf	A&H	Budinich	A&H	LRB	T&K
`brock200-1`	21	149	98	100	172	157	56
`brock200-2`	12	100	95	100	140	105	35
`brock200-3`	15	121	98	100	155	127	43
`brock200-4`	17	132	96	100	162	139	48
`C125.9`	34	112	63	62	118	115	55
`C250.9`	44	224	124	125	237	230	98
`c-fat200-1`	12	17	88	181	53	136	15
`c-fat200-2`	24	33	95	192	79	129	24
`c-fat200-5`	58	85	87	196	130	107	84
`c-fat500-1`	14	20	244	428	90	347	14
`c-fat500-10`	126	187	221	477	305	280	126
`c-fat500-2`	26	38	217	449	132	340	26
`c-fat500-5`	64	93	222	453	214	319	64
`DSJC500-5`	13	251	245	250	353	259	70
`gen200-p0.9-44`	44	180	96	100	189	184	62
`gen200-p0.9-55`	55	180	97	100	189	184	72
`hamming6-2`	32	58	33	42	60	59	37
`hamming6-4`	4	22	28	32	37	36	10
`hamming8-2`	128	248	131	163	251	249	136
`hamming8-4`	16	163	121	128	204	173	34
`johnson16-2-4`	8	92	16	60	104	96	14
`johnson8-2-4`	4	15	8	14	20	16	6
`johnson8-4-4`	14	53	28	35	61	56	19
`keller4`	11	111	63	85	137	117	34
`MANN-a9`	16	41	20	22	43	42	21
`p-hat300-1`	8	80	139	150	147	184	24
`p-hat300-2`	25	158	145	150	209	182	46
`p-hat300-3`	36	225	147	150	258	235	73
`p-hat500-1`	9	137	238	250	250	305	36
`p-hat500-2`	36	272	243	250	354	308	70
`p-hat700-1`	11	190	335	350	348	429	44
`p-hat700-2`	44	377	343	350	493	430	91
`san200-0.7-1`	30	140	95	100	167	148	30
`san200-0.7-2`	18	143	77	100	167	148	18
`san200-0.9-1`	70	180	98	100	189	184	70
`san200-0.9-2`	60	180	98	100	189	184	60
`san200-0.9-3`	44	180	96	100	189	184	44
`san400-0.5-1`	13	202	176	200	282	212	13
`sanr200-0.7`	18	139	97	100	166	147	50
`sanr200-0.9`	42	179	99	100	189	184	79
`sanr400-0.5`	13	201	196	200	282	208	59

Table 3.3: CPU times (in milliseconds) for the considered bounds on the clique number.

	Spectral			Linear		Coloring
Name	Wilf	A&H	Budinich	A&H	LRB	T&K
`brock200-1`	29.687	23.437	23.437	0.000	0.000	3.125
`brock200-2`	25.000	23.437	21.875	0.000	0.000	3.125
`brock200-3`	23.437	23.437	20.312	1.562	0.000	1.562
`brock200-4`	23.437	23.437	23.437	0.000	1.562	1.562
`C125.9`	7.812	6.250	7.812	0.000	0.000	0.000
`C250.9`	40.625	43.750	40.625	0.000	0.000	3.125
`c-fat200-1`	17.187	17.187	21.875	0.000	0.000	0.000
`c-fat200-2`	10.937	12.500	23.437	0.000	1.562	1.562
`c-fat200-5`	15.625	17.187	25.000	0.000	0.000	3.125
`c-fat500-1`	151.563	142.188	232.813	1.562	0.000	6.250
`c-fat500-10`	168.750	164.063	292.188	0.000	0.000	32.812
`c-fat500-2`	142.188	142.188	273.438	1.562	0.000	7.812
`c-fat500-5`	137.500	139.063	245.313	1.562	0.000	17.187
`DSJC500-5`	267.188	260.938	257.813	4.688	3.125	17.187
`gen200-p0.9-44`	23.437	21.875	21.875	1.562	0.000	1.562
`gen200-p0.9-55`	23.437	23.437	21.875	1.562	0.000	1.562
`hamming6-2`	1.562	1.562	1.562	0.000	0.000	0.000
`hamming6-4`	1.562	1.562	1.562	0.000	0.000	0.000
`hamming8-2`	42.187	42.187	43.750	0.000	1.562	21.875
`hamming8-4`	42.187	40.625	39.062	0.000	1.562	14.062
`johnson16-2-4`	4.687	4.687	6.250	0.000	0.000	3.125
`johnson8-2-4`	0.000	0.000	0.000	0.000	0.000	0.000
`johnson8-4-4`	1.562	1.562	1.562	0.000	0.000	0.000
`keller4`	17.187	18.750	20.312	0.000	0.000	3.125
`MANN-a9`	0.000	1.562	0.000	0.000	0.000	0.000
`p-hat300-1`	64.062	64.062	64.062	1.562	0.000	4.687
`p-hat300-2`	65.625	65.625	68.750	1.562	1.562	4.687
`p-hat300-3`	67.187	67.187	65.625	1.562	0.000	6.250
`p-hat500-1`	262.500	264.063	257.813	1.562	3.125	14.062
`p-hat500-2`	264.063	262.500	262.500	3.125	3.125	15.625
`p-hat700-1`	667.188	667.188	675.000	3.125	4.687	29.687
`p-hat700-2`	668.750	667.188	667.188	6.250	6.250	29.687
`san200-0.7-1`	20.312	23.437	23.437	0.000	1.562	3.125
`san200-0.7-2`	35.937	34.375	37.500	0.000	0.000	4.687
`san200-0.9-1`	23.437	21.875	23.437	0.000	0.000	4.687
`san200-0.9-2`	21.875	25.000	23.437	0.000	0.000	3.125
`san200-0.9-3`	23.437	23.437	25.000	0.000	0.000	3.125
`san400-0.5-1`	135.938	143.750	135.938	1.562	1.562	18.750
`sanr200-0.7`	23.437	23.437	23.437	0.000	1.562	1.562
`sanr200-0.9`	25.000	21.875	23.437	0.000	0.000	1.562
`sanr400-0.5`	140.625	139.063	142.188	1.562	3.125	10.937

MANN, and p-hat families and applied an upper time limit of three hours to solve each instance. Shimizu et al. [72] report results for 34 of the DIMACS instances, with a 1,000-second time limit. Our experiment involves all families of DIMACS-EW graphs. In addition to all the graphs with up to 300 vertices, we consider instances with up to 700 vertices with edge density no larger than 0.50 and use a three-hour time constraint for each instance.

Table 3.4 summarizes the recorded solution time. In this table, W^* shows the optimal solution value of the MEWC problem (if known) for each instance. The column "IP" provides the results of [25]. They report the solution time of nine different integer programming models for the MEWC problem using the ILOG/CPLEX 11.2 solver on an Intel® Core i-7 CPU @3.40 GHz and 8 GB RAM. We consider the best model for each instance among them in our comparison. The column "CBQ" shows the computational results of the algorithm presented in [2], using the same system used to run the algorithms proposed in this chapter. The column "EWC" presents the results for EWCLIQUE algorithm reported in [72]. The results for EWC were copied from [72] and were obtained using a computer running Linux 4.4.0 OS with Intel® Core i-7 CPU @3.40 GHz and 16 GB RAM. Shimizu et al. [72] implemented their EWCLIQUE algorithm in C++ and used g++ 5.4.0 compiler with optimization option -O2. The last two columns concern the results of the algorithm presented in this chapter. The column "Alg.1" shows the solution time when the SUBCOLOR^1 function was used in the algorithm and "Alg.2" corresponds to application of SUBCOLOR^2. We have used "t-lim" in this table to indicate that the corresponding algorithm was terminated due to time limit before completing the search. We have run our algorithm with the initial lower bound used in [2]. The reported results in Table 3.4 include the computing time of the initial lower bounds.

We could solve 36 out of 41 instances within three hours. The only instance (among successful ones) that took more than one hour for our method to solve was san200-0.9-3. Our solution times are an order of magnitude better than the best results obtained by IP and CBQ methods for instances considered by all the three methods (except for small graphs which are solved quickly by all methods). In comparison of our method with EWCLIQUE, neither of them performs uniformly

Table 3.4: Solution time for the MEWC problem.

		CPU (sec.)				
Name	W^*	IP	CBQ	EWC	Alg.1	Alg.2
brock200-1	21,230	t-lim	3,047.565	338.31	196.359	282.281
brock200-2	6,542	9,464.240	7.436	0.10	0.765	0.983
brock200-3	10,303	t-lim	55.905	1.27	5.140	6.733
brock200-4	13,967	t-lim	188.031	4.84	16.311	21.765
C125.9	66,248	t-lim	4,558.170	-	108.109	173.719
C250.9	-	t-lim	t-lim	-	t-lim	t-lim
c-fat200-1	7,734	3.870	0.483	< 0.01	0.030	0.046
c-fat200-2	26,389	33.260	0.890	< 0.01	0.046	0.061
c-fat200-5	168,200	155.300	t-lim	74.31	0.077	0.124
c-fat500-1	10,738	-	-	< 0.01	0.343	0.390
c-fat500-10	804,000	-	-	t-lim	723.406	1,357.550
c-fat500-2	38,350	-	-	< 0.01	0.374	0.421
c-fat500-5	205,864	-	-	0.43	0.702	0.905
DSJC500-5	9,626	-	-	44.43	204.109	254.406
gen200-p0.9-44	-	-	-	-	t-lim	t-lim
gen200-p0.9-55	150,839	-	-	-	286.140	429.234
hamming6-2	32,736	0.300	4.437	< 0.01	0.015	0.031
hamming6-4	396	1.970	0.031	< 0.01	0.000	0.000
hamming8-2	800,624	t-lim	t-lim	0.23	t-lim	t-lim
hamming8-4	12,360	t-lim	439.437	1.46	14.390	18.421
johnson16-2-4	3,808	t-lim	84.687	0.25	12.203	15.984
johnson8-2-4	192	0.140	0.000	< 0.01	0.000	0.000
johnson8-4-4	6,552	2.340	0.687	< 0.01	0.046	0.078
keller4	6,745	t-lim	42.218	0.70	2.233	2.952
MANN-a9	5,460	9.390	1.906	0.02	0.203	0.359
p-hat300-1	3,321	1,273.050	3.281	0.01	0.327	0.374
p-hat300-2	31,564	t-lim	171.281	42.90	10.468	14.984
p-hat300-3	63,390	t-lim	t-lim	-	3,576.471	5,152.831
p-hat500-1	4,764	-	-	0.13	2.296	2.671
p-hat500-2	63,870	-	-	-	1,023.799	1,519.049
p-hat700-1	5,185	-	-	0.52	9.280	10.921
p-hat700-2	-	-	-	-	t-lim	t-lim
san200-0.7-1	45,295	-	-	54.88	1.827	2.421
san200-0.7-2	15,073	-	-	17.86	0.718	0.858
san200-0.9-1	242,710	-	-	12.56	89.249	124.234
san200-0.9-2	178,468	-	-	833.49	160.765	215.656
san200-0.9-3	96,764	-	-	-	4,037.105	8,740.675
san400-0.5-1	7,442	-	-	60.36	11.530	15.187
sanr200-0.7	16,398	-	-	18.67	45.233	62.936
sanr200-0.9	-	-	-	-	t-lim	t-lim
sanr400-0.5	8,298	-	-	9.04	52.827	66.046

better than the other. However, on the 30 instances solved by both methods, the total time to solve all these 30 instances taken by Alg.1, Alg.2, and EWCLIQUE was 837.866, 1,122.228, and 1,516.54 seconds, respectively. The difference is mainly due to more challenging instances of this set. In fact, for most instances that the reported solution time of EWCLIQUE is less than Alg.1, the solution time for both algorithms is less than 10 seconds. Regarding the two vertex coloring routines employed in our method, the algorithm performed uniformly better when the method of Tomita and Kameda [49] was used in the pruning process. The difference is noticeable for instances that took more time to solve; see for example the results corresponding to `san200-0.9-3`.

Our best time improvements were obtained for `c-fat200-5`, `C125.9` and `san` family instances. The CBQ algorithm failed to solve `c-fat200-5` within three hours while it was solved by one of the integer programming models of Gouveia and Martins [25] in about 155 seconds, and by EWCLIQUE in 74.31 seconds. We solved this instance in 0.077 seconds. All nine integer programming models of Gouveia and Martins [25] failed to solve `C125.9`, while we solved this instance 42 times faster than the CBQ algorithm. For the five instances of the `san` family considered by our method and EWCLIQUE, the total time taken by Alg.1 to solve all these five instances was 264.089 seconds, while it took 979.15 seconds for EWCLIQUE to solve them. It should be noted that both Shimizu et al. [72] and Gouveia and Martins [25] used computers with slightly faster processors (3.40 GHz) than that of the machine we used for our experiments (2.90 GHz).

Among the five solution methods in Table 3.4, four are combinatorial B&B algorithms, three of which (EWCLIQUE being an exception) use the same initial lower bounds and vertex ordering. For each of these methods, we present the number of nodes of the corresponding search tree in Table 3.5. We exclude from this table the five instances that we could not solve. As it can be seen in this table, the number of B&B nodes in either version of the algorithm presented in this paper is much less than that of the CBQ and EWCLIQUE algorithms, with one exception for each CBQ and EWCLIQUE. Namely, CBQ uses less nodes for `johnson8-2-4`, which is the smallest graph among the test instances; and EWCLIQUE uses less nodes for `hamming6-4`. Besides, the number of B&B nodes under the second chromatic method in our algorithm is less than (or equal to) the first method for all but one of the considered instances, `san200-0.9-3`. Along with the results of Table 3.4, this implies that better quality of the upper bound (on the clique number) generated by the second chromatic method did not make up for the overhead of calculating the pruning threshold for the MEWC problem.

Table 3.5: Size of the search tree.

	Number of B&B nodes			
Name	CBQ	EWC	Alg.1	Alg.2
brock200-1	19,445,372	1,328,614,116	3,058,744	2,853,358
brock200-2	54,328	345,371	14,418	13,683
brock200-3	443,026	4,282,305	99,441	92,784
brock200-4	1,357,862	13,814,425	284,391	264,588
C125.9	13,276,576	-	1,022,493	1,018,056
c-fat200-1	186	632	186	186
c-fat200-2	2,431	6,780	178	178
c-fat200-5	-	138,193,445	176	176
c-fat500-1	-	1,605	486	486
c-fat500-10	-	-	586,791	586,791
c-fat500-2	-	4,679	472	472
c-fat500-5	-	1,227,023	750	750
DSJC500-5	-	200,152,687	2,994,499	2,856,281
gen200-p0.9-55	-	-	572,542	520,760
hamming6-2	16,134	896	223	213
hamming6-4	403	340	351	351
hamming8-4	2,568,843	2,475,100	155,100	145,634
johnson16-2-4	1,544,956	1,905,154	1,189,751	1,173,959
johnson8-2-4	113	150	127	127
johnson8-4-4	5,532	3,953	1,407	1,360
keller4	353,436	2,158,496	41,474	40,271
MANN-a9	40,407	116,041	26,159	26,159
p-hat300-1	11,278	50,151	3,939	3,862
p-hat300-2	854,903	134,486,327	114,726	110,475
p-hat300-3	-	-	23,035,890	21,907,679
p-hat500-1	-	468,371	25,159	23,945
p-hat500-2	-	-	6,211,268	5,893,921
p-hat700-1	-	1,678,557	113,239	108,136
san200-0.7-1	-	387,149,894	6,717	5,917
san200-0.7-2	-	48,732,878	3,983	3,765
san200-0.9-1	-	12,731,307	219,755	174,470
san200-0.9-2	-	303,169,816	377,156	303,253
san200-0.9-3	-	-	11,876,914	18,700,526
san400-0.5-1	-	43,132,933	41,090	40,953
sanr200-0.7	-	55,871,909	814,221	754,629
sanr400-0.5	-	36,003,126	928,826	881,902

4. POLYHEDRAL PROPERTIES OF THE INDUCED CLUSTER SUBGRAPHS*

4.1 Introduction

A graph is called a *cluster graph* if its every connected component is a complete graph. Given a simple, undirected graph $G = (V, E)$, with the set of vertices V and the set of edges E, an *independent union of cliques (IUC)* is a subset of vertices $U \subseteq V$ inducing a cluster graph. The *maximum IUC problem* is to find an IUC of maximum cardinality in G. The importance of this problem stems from its application in *graph clustering* (partitioning the vertices of a graph into cohesive subgroups), which is a fundamental task in unsupervised data analysis; see [74] and the references therein for a survey of graph clustering applications and methods. A cluster graph is an ideal instance from the standpoint of cluster analysis—as it is readily composed of mutually disjoint "tightly knit" subgroups—and the maximum IUC problem aims to identify the largest induced cluster graph contained in an input graph G. Such an induced subgraph uncovers important information about the heterogeneous structure of the graph, including the inherent number of its clusters and their cores (centers). Besides the practical importance of this problem, the definition of IUC subsumes two fundamental structures in a graph, i.e., cliques and independent sets. The *maximum clique problem* and *maximum independent set problem* are among the most popular problems of combinatorial optimization, several variants of which have been studied in the literature; see e.g. [3, 55, 23, 75, 76, 77]. Therefore, the study of the maximum IUC problem, that admits both of these complementary structures as feasible solutions, is also of a particular theoretical interest.

The maximum IUC problem has appeared under different names in the literature. Fomin et al. [78] introduced this problem as the *maximum induced cluster subgraph problem*, and proposed an exact $\mathcal{O}(1.6181^n n^{\mathcal{O}(1)})$-time algorithm for it, where $n = |V|$. Ertem et al. [79] considered the maximum IUC problem as a relaxation of the *maximum α-cluster problem* with the maximum local clustering coefficient, i.e., $\alpha = 1$. They proposed a graph clustering algorithm based on

* This chapter is reprinted with permission from "Polyhedral properties of the induced cluster subgraphs" by S. Hosseinian, and S. Butenko [73]. Copyright © 2021, Elsevier B.V.

a given maximum IUC, referred to as disjoint 1-clusters in [79], and showed the competence of their method via experiments with real-life social networks. Even though the term "independent union of cliques" was used earlier (see, e.g., [80]), it was not until very recently that the corresponding optimization problem was referred to as "MAXIMUM IUC" by Ertem et al. [81]. In this work, the authors studied some basic properties of IUCs and analyzed complexity of the maximum IUC problem on some restricted classes of graphs. They also performed computational experiments using a combinatorial algorithm (Russian Doll Search) and an integer (linear) programming formulation of this problem. To the best of our knowledge, this is the only work containing computational experiments on the maximum IUC problem, and their results show that this problem is quite challenging for the exact solution methods. It is known that a graph is a cluster graph if and only if it contains no induced subgraph isomorphic to P_3 (the path graph on three vertices) [82]. In this regard, the maximum IUC problem has also been referred to as the *maximum induced P_3-free subgraph problem* [78]. In our presentation to follow, we refer to induced subgraphs isomorphic to P_3 as *open triangles*.

Evidently, finding a maximum IUC in G is equivalent to finding a minimum number of vertices whose deletion turns G into a cluster graph. The latter is the optimization version of the *cluster vertex deletion problem (k-CVD)*, which asks if an input graph G can be transformed into a cluster graph by deleting at most k vertices, and is known to be NP-hard by Lewis and Yannakakis theorem [83]. The study of this problem started from the viewpoint of parameterized complexity with the work of Gramm et al. [82], who proposed an $\mathcal{O}(2.26^k + nm)$-time fixed-parameter tractable algorithm for k-CVD, where $n = |V|$ and $m = |E|$. Later, Hüffner et al. [84] and Boral et al. [85] improved this result to $\mathcal{O}(2^k k^9 + nm)$ and $\mathcal{O}(1.9102^k(n+m))$, respectively. Le et al. [86] showed that k-CVD admits a kernel with $\mathcal{O}(k^{5/3})$ vertices, which means there exists a polynomial-time algorithm that converts an n-vertex instance of k-CVD to an equivalent instance with $\mathcal{O}(k^{5/3})$ vertices. Fomin et al. [87] used the parameterized results of [85] to show that the minimum CVD (hence, maximum IUC) problem is solvable in $\mathcal{O}(1.4765^{n+o(n)})$ time, which is better than the former result of [78]. It is also known that the minimum CVD problem is approximable to within a constant factor. In particular, You et al. [88] proposed a $\frac{5}{2}$-approximation algorithm for it, and Fiorini et al. [89] improved the approximation ratio to $\frac{7}{3}$. Several other problems involving cluster subgraphs (not necessarily induced) have also been considered in the literature, including *cluster editing* [90, 91, 92, 93, 94, 95, 96], *disjoint cliques* [97, 98], *s-plex cluster vertex deletion* [99], and *s-plex cluster editing* [100].

From a broader perspective, the maximum IUC problem is to identify a maximum-cardinality *independent set* in a graph-based *independence system*. An independence system $(S, \mathcal{S})$ is a pair of a finite set S together with a family $\mathcal{S}$ of subsets of S such that $I_2 \subseteq I_1, I_1 \in \mathcal{S}$ implies $I_2 \in \mathcal{S}$. Referring to an independence system $(S, \mathcal{S})$, every $I \in \mathcal{S}$ is called an independent set and every $D \subseteq S, D \notin \mathcal{S}$, is called a *dependent set*. The minimal (inclusion-wise) dependent subsets of S are called *circuits* of the independence system $(S, \mathcal{S})$. In graph-theory terminology, an independent set (stable set, vertex packing) is a subset of pairwise non-adjacent vertices in a graph $G = (V, E)$. Let $\mathcal{I}$ denote the set of all independent sets (in the sense of graph theory) in G. Then, $(V, \mathcal{I})$ is an independence system whose circuit set is given by E; hence, the two definitions coincide. Following this concept, V together with the set of all IUCs in G form an independence system whose set of circuits is given by the three-vertex subsets of V inducing open triangles in the graph. This perspective on the maximum IUC problem is of our interest as some of the results to follow correspond to certain properties of independence systems. It is worth noting that many well-known problems of graph theory, beyond those mentioned here, convey the notion of independence systems associated with a graph; see for example [101, 102].

Here, we present a comprehensive study of the maximum IUC problem from the standpoint of polyhedral combinatorics. More specifically, we study the facial structure of the IUC polytope associated with a simple undirected graph G, and identify several facet-defining valid inequalities for this polytope. As a result, we will be able to present the full description of the IUC polytope for some special classes of graphs. We also study the computational complexity of the separation problems for these inequalities, and perform computational experiments to examine their effectiveness when used in a branch-and-cut scheme to solve the maximum IUC problem.

4.1.1 Terminology and notation

Throughout this chapter, we consider a simple undirected graph $G = (V, E)$, where V is the vertex set and $E \subseteq \{\{i, j\} \mid i, j \in V, i \neq j\}$ is the edge set. Unless otherwise stated, we assume $|V| = n$. Given a subset of vertices $S \subseteq V$, the subgraph *induced* by S is denoted by $G[S]$ and is defined as $G[S] = (S, E(S))$, where $E(S)$ is the set of edges with both endpoints in S. A *clique* is a subset of vertices inducing a complete graph, and an *independent set* (stable set, vertex packing) is a subset of vertices inducing an edgeless graph. A clique (resp. independent set) is called *maximum* if it is of maximum cardinality, i.e., there is no larger clique (resp. independent set) in the graph. Cardinality of a maximum clique in G is called the *clique number* of the graph, and is denoted by $\omega(G)$. Respectively, the *independence number* of G is the cardinality of a maximum independent set in the graph, and is denoted by $\alpha(G)$. In a similar manner, a *maximum* IUC in G is an IUC of maximum cardinality, and the *IUC number* of G, denoted by $\alpha^\omega(G)$, is the cardinality of a maximum IUC in the graph. Since every clique as well as every independent set in G is an IUC, the IUC number of the graph is bounded from below by its clique number and independence number, i.e., $\max\{\omega(G), \alpha(G)\} \leq \alpha^\omega(G)$. The (open) *neighborhood* of a vertex $i \in V$ is the set of vertices $j \in V \backslash \{i\}$ adjacent to i, and is denoted by $N(i)$. The *closed neighborhood* of i is defined as $N[i] = N(i) \cup \{i\}$. The *incidence vector* of a subset of vertices $S \subseteq V$ is a binary vector $x \in \mathbb{R}^n$ such that $x_i = 1, \forall i \in S$, and $x_i = 0, \forall i \in V \backslash S$. Conventionally, the incidence vector of a single vertex $i \in V$ is denoted by e_i. We use $\mathbf{0}$, $\mathbf{1}$ and $\mathbf{2}$ to denote the vectors of all zero, one and two, respectively. The corresponding dimensions will be clear from the context.

We assume familiarity of the reader with fundamentals of polyhedral theory, and just briefly mention the concept of *lifting* [103, 104, 105], as several proofs to follow are based on lifting arguments. Let $\mathscr{P}$ be the convex hull of the set of 0-1 feasible solutions to an arbitrary system of linear inequalities with a nonnegative coefficient matrix, i.e., $\mathscr{P} = \text{conv}\{x \in \{0, 1\}^n \mid Ax \leq b\}$. Corresponding to a subset of variables indexed by $N' \subseteq N = \{1, \dots, n\}$, consider the polytope $\mathscr{P}_{(x_{N'}=0)}$ obtained from $\mathscr{P}$ by setting $x_j = 0, \forall j \in N'$. Let $N'' = N \backslash N'$, and suppose that the inequality $\sum_{i \in N''} \pi_i x_i \leq \pi_0$ induces a facet of $\mathscr{P}_{(x_{N'}=0)}$. Then, lifting a variable $x_j, j \in N'$, into this inequality leads to an inequality $\pi_j x_j + \sum_{i\in N''} \pi_i x_i \leq \pi_0$, where

$$\pi_j = \pi_0 - \max\left\{\sum_{i\in N''} \pi_i x_i | \sum_{i\in N''} A_i x_i \leq b - A_j;\ x_i \in \{0,1\}, \forall i \in N''\right\}, \tag{4.1}$$

which is facet-defining for $\mathscr{P}_{(x_{N'\setminus\{j\}}=0)}$. In (4.1), A_i denotes the coefficient vector of a variable x_i in the system of inequalities, i.e., the i-th column of A. The importance of this result is due to the fact that some facets of the original polytope $\mathscr{P}$ can be obtained from the facets of the restricted polytopes through sequential (or simultaneous) lifting procedures. In particular, some facets of the polytope associated with a graph G for a certain problem may be identified through lifting the facets of the lower-dimensional polytopes corresponding to its subgraphs. Finally, note that (4.1) is equivalent to

$$\pi_j = \pi_0 - \max\left\{\sum_{i\in N''} \pi_i x_i \mid x \in \mathscr{P} \text{ and } x_j = 1\right\}, \tag{4.2}$$

which is easier to use when $\mathscr{P}$ is defined to be the convex hull of the incidence vectors of the sets of vertices in a graph holding a certain property.

4.2 The IUC Polytope

The IUC polytope associated with a simple, undirected graph G, denoted by $\mathscr{P}_{\mathcal{IUC}}(G)$, is the convex hull of the incidence vectors of all IUCs in G, which can be characterized by the circuit set of the corresponding independence system as follows [106]:

$$\mathscr{P}_{\mathcal{IUC}}(G) = \text{conv}\{x \in \{0,1\}^n \mid x_i + x_j + x_k \leq 2, \forall \{i,j,k\} \in \Lambda(G)\}, \tag{4.3}$$

where $\Lambda(G)$ denotes the set of three-vertex subsets of V inducing open triangles in G. Patently, $\alpha^\omega(G) = \max\{\sum_{i\in V} x_i \mid x \in \mathscr{P}_{\mathcal{IUC}}(G)\}$, and the *maximum weight IUC* problem, which is to find an IUC with the maximum total weight in a vertex-weighted graph, can be formulated as

$$\text{Maximize} \sum_{i\in V} w_i x_i \ \text{ subject to } \ x \in \mathscr{P}_{\mathcal{IUC}}(G), \tag{4.4}$$

with w_i denoting the weight of a vertex $i \in V$. We call each inequality $x_i + x_j + x_k \leq 2, \{i,j,k\} \in \Lambda(G)$, an open-triangle (OT) inequality. The following theorems establish the basic properties of $\mathscr{P}_{\mathcal{IUC}}(G)$ including the strength of the OT inequalities.

Theorem 7. *$\mathscr{P}_{\mathcal{IUC}}(G)$ is full-dimensional, and for every $i \in V$, the inequalities $x_i \geq 0$ and $x_i \leq 1$ are facet-defining for this polytope.*

Proof. Note that $\mathbf{0} \in \mathscr{P}_{\mathcal{IUC}}(G)$. Also, every single vertex, as well as every pair of vertices in G is an IUC, by definition. Full-dimensionality of $\mathscr{P}_{\mathcal{IUC}}(G)$ is established by noting that $\mathbf{0}$ and $e_i, \forall i \in V$, are $n+1$ affinely independent points in $\mathbb{R}^n$ belonging to this polytope. Besides, for every $i \in V$, $\mathbf{0}$ and $e_j, \forall j \in V \backslash \{i\}$, belong to $\mathcal{F}_i^0 = \{x \in \mathscr{P}_{\mathcal{IUC}}(G) \mid x_i = 0\}$, which indicates that $\mathcal{F}_i^0$ is a facet of $\mathscr{P}_{\mathcal{IUC}}(G)$. Similarly, for every $i \in V$, the points e_i and $e_{ij} = e_i + e_j, \forall j \in V \backslash \{i\}$, are affinely independent and belong to $\mathcal{F}_i^1 = \{x \in \mathscr{P}_{\mathcal{IUC}}(G) \mid x_i = 1\}$, hence $\mathcal{F}_i^1$ is a facet of $\mathscr{P}_{\mathcal{IUC}}(G)$. □

It follows from full-dimensionality of $\mathscr{P}_{\mathcal{IUC}}(G)$ that every facet-defining inequality of this polytope is unique to within a positive multiple. Thus, in order to prove that a proper face $\mathcal{F}$ of $\mathscr{P}_{\mathcal{IUC}}(G)$ defined by a valid inequality $\pi(x) \leq \pi_0$ is a facet, it suffices to show that any supporting hyperplane of $\mathscr{P}_{\mathcal{IUC}}(G)$ containing $\mathcal{F}$ is a (non-zero) scalar multiple of $\pi(x) = \pi_0$.

Theorem 8. *The inequality $x_i + x_j + x_k \leq 2, \{i,j,k\} \in \Lambda(G)$, is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$ if and only if there does not exist a vertex $v \in V \backslash \{i,j,k\}$ such that $H = \{i,j,k,v\}$ induces a chordless cycle in G.*

Proof. Clearly, $\mathcal{F} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G) \mid x_i + x_j + x_k = 2\}$ is a proper face of $\mathscr{P}_{\mathcal{IUC}}(G)$. So, suppose that it is contained in a supporting hyperplane $\mathscr{H} : \sum_{v \in V} a_v x_v = b$ of this polytope. Observe that,

$$e_i + e_j \in \mathcal{F} \subseteq \mathscr{H} \quad \therefore \quad a_i + a_j = b.$$

Similarly, $a_i + a_k = b$ and $a_j + a_k = b$, which imply $a_i = a_j = a_k = a$ and $b = 2a$. Now, consider $G[H]$ the subgraph induced by $H = \{i,j,k,v\}$ for an arbitrary vertex $v \in V \backslash \{i,j,k\}$. As $G[H]$ is not a chordless cycle, it contains at most three open triangles, and those have at least one vertex in common. Let k be such a vertex, then

$$e_i + e_j + e_v \in \mathcal{F} \subseteq \mathscr{H} \;\; \therefore \;\; a_i + a_j + a_v = b \;\; \therefore \;\; a_v = 0, \; \forall v \in V \backslash \{i, j, k\}.$$

As the interior of $\mathscr{P}_{\mathcal{IUC}}(G)$ is nonempty, $a \neq 0$ and $\mathscr{H}$ is a (non-zero) scalar multiple of $x_i + x_j + x_k = 2$. To complete the proof, note that if $H = \{i, j, k, v\}$ induces a chordless cycle in G, then $x_i + x_j + x_k \leq 2$ is dominated by the inequality $x_i + x_j + x_k + x_v \leq 2$, which is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$ by Theorem 9 of the next section. $\square$

In Section 4.3 we will show that Theorem 8 is a special case of a more general statement for the so-called *star inequality*, defined later.

Obviously, the maximum IUC problem can be directly solved using off-the-shelf MIP solvers through the following formulation:

$$\begin{aligned} \alpha^{\omega}(G) \;=\; \max \quad & \sum_{i \in V} x_i \\ \text{s.t.} \quad & x_i + x_j + x_k \leq 2, \quad \forall \{i, j, k\} \in \Lambda(G), \\ & x_i \in \{0, 1\}, \qquad\quad \forall\, i \in V. \end{aligned} \tag{4.5}$$

However, the LP relaxation of (4.5) is generally too weak. In fact, given the feasibility of the fractional assignment $x_i = \frac{2}{3}, \forall i \in V$, the optimal solution value of the LP relaxation problem is at least as large as $\frac{2n}{3}$, while the computational results of Ertem et al. [81] show that the IUC number of graphs with moderate densities tends to stay in a close range of their relatively small independence and clique numbers. This is due to the fact that the fractional IUC polytope $\mathscr{P}^F_{\mathcal{IUC}}(G) = \{x \in [0, 1]^n \mid x_i + x_j + x_k \leq 2, \forall \{i, j, k\} \in \Lambda(G)\}$ is a polyhedral relaxation of a *cubically* constrained region in the space of original variables. Note that the IUC property may be enforced through trilinear products of the variables in a 0-1 program, i.e., by replacing the OT inequalities of (4.5) with $x_i x_j x_k = 0, \forall \{i, j, k\} \in \Lambda(G)$. In such a cubic formulation, relaxing the integrality of variables will not change the set of optimal solutions, because given a set of variables $x_i = 0, \forall i \in V_0 \subset V$, to satisfy the constraints, the optimality conditions necessitate $x_i = 1, \forall i \in V \backslash V_0$. Hence, $\alpha^\omega(G)$ can be obtained by solving the following continuous problem:

$$\alpha^\omega(G) = \max_{x \in [0,1]^n} \left\{ \sum_{i \in V} x_i \mid x_i x_j x_k = 0, \forall \{i,j,k\} \in \Lambda(G) \right\}. \tag{4.6}$$

It is known that the (convex and concave) envelopes of a trilinear monomial term $w = xyz$ when $0 \leq x, y, z \leq 1$ are as follows [107]:

$$w \leq x, \quad w \leq y, \quad w \leq z, \quad w \geq x + y + z - 2, \quad w \geq 0,$$

which, along with $w_{ijk} = x_i x_j x_k = 0, \forall \{i,j,k\} \in \Lambda(G)$, indicates that $\mathscr{P}^F_{\mathcal{IUC}}(G)$ is a polyhedral outer-approximation of the feasible region of (4.6). As a result, the gap between $\alpha^\omega(G)$ and the optimal solution value of the LP relaxation problem of (4.5) is due to accumulation of the gap between the trilinear terms and their overestimating envelopes. A similar result holds between continuous and integer (linear) programming formulations of the maximum independent set and maximum clique problems, except that the corresponding continuous formulations of those problems are quadratically constrained. The cubic nature of the IUC formulation justifies the weakness of $\mathscr{P}^F_{\mathcal{IUC}}(G)$ even compared to the fractional independent set (vertex packing) and clique polytopes, which are known to be loose in general. This further reveals the importance of exploring the facial structure of the IUC polytope and applying strong cutting planes in integer (linear) programming solution methods of the maximum IUC problem.

4.3 Facet-producing Structures

In this section, we present some facet-inducing valid inequalities for the IUC polytope associated with chordless cycle (and its edge complement), star (and double-star), fan and wheel graphs. We study the conditions under which these inequalities remain facet-defining for the IUC polytope of an arbitrary graph that contains the corresponding structure as an induced subgraph, and present some results with respect to lifting procedure for those that do not have this property. In some cases, we will present the full description of the IUC polytope associated with these graphs.

4.3.1 Chordless cycle and its complement

The following theorem concerns the facial structure of the IUC polytope associated with a (chordless) cycle graph. In an arbitrary graph $G = (V, E)$, a subset of vertices $H \subseteq V,\ |H| \geq 4$, inducing a chordless cycle is called a *hole*. In labeling the vertices of a hole H, we assume that adjacent vertices have consecutive labels with the convention $|H| + 1 \equiv 1$.

Theorem 9. **(Hole Inequality)** *Let $H \subseteq V$ be a hole of cardinality $|H| = 3q + r$ in $G = (V, E)$, where q is a positive integer and $r \in \{0, 1, 2\}$. Then, the inequality*

$$\sum_{i \in H} x_i \leq 2q + \left\lfloor \frac{2r}{3} \right\rfloor \tag{4.7}$$

is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$. Moreover, (i) *it induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[H])$ if and only if $r \neq 0$, and* (ii) *it is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$ if $q = r = 1$.*

Proof. Consider $G[H]$ the subgraph induced by H. This subgraph contains $|H|$ distinct open triangles, and each vertex $i \in H$ appears in exactly three of them. Summation of the corresponding OT inequalities leads to

$$3 \sum_{i \in H} x_i = \sum_{\{i,j,k\} \in \Lambda(G[H])} x_i + x_j + x_k \leq 2|H| = 6q + 2r.$$

After dividing both sides of the last inequality by 3, integrality of the left-hand side implies validity of (4.7) for $\mathscr{P}_{\mathcal{IUC}}(G)$ as a Gomory-Chvátal cut.

(i) It is clear that, as a linear combination of OT inequalities, (4.7) is not facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G[H])$ if $r = 0$. Thus, we just need to show that $r \neq 0$ is sufficient for (4.7) to induce a facet of this polytope. We examine the corresponding cases separately.

- CASE 1: $r = 1$

  Consider a sequential partitioning of H such that each partition contains exactly three vertices except for the last one, which is left with a single vertex. Assume a labeling of the vertices such that the first partition is given by $\{1,2,3\}$, and let x_{U_1} and x_{U_2} be the incidence vectors of the following sets of vertices:

$$U_1 = \{1, 3, 3p-1, 3p, \forall p \in \{2, \ldots, q\}\},$$
$$U_2 = \{2, 3, 3p-1, 3p, \forall p \in \{2, \ldots, q\}\}.$$

The sets of black vertices in Figures 4.1(a) and 4.1(b) depict U_1 and U_2, respectively.

Observe that, $x_{U_1}, x_{U_2} \in \mathcal{F} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G[H]) \mid \sum_{v\in H} x_v = 2q\}$, hence $\mathcal{F}$ is a proper face of $\mathscr{P}_{\mathcal{IUC}}(G[H])$. Suppose that $\mathscr{H} : \sum_{v\in H} a_v x_v = b$ is an arbitrary supporting hyperplane of $\mathscr{P}_{\mathcal{IUC}}(G[H])$ containing $\mathcal{F}$. Then, $x_{U_1}, x_{U_2} \in \mathscr{H}$ implies that $a_1 = a_2$. The same argument using similar partitions can be used to show that this result holds for every two adjacent vertices in $G[H]$, thus $a_v = a, \forall v \in H$, and $b = 2qa$. The proof is complete by noting that the interior of $\mathscr{P}_{\mathcal{IUC}}(G[H])$ is nonempty, so $a \neq 0$.

- CASE 2: $r = 2$

  Similar to the former case, let x_{U_1} and x_{U_2} be the incidence vectors of

$$U_1 = \{3q+2, 1, 3, 3p-1, 3p, \forall p \in \{2, \ldots, q\}\},$$
$$U_2 = \{3q+2, 2, 3, 3p-1, 3p, \forall p \in \{2, \ldots, q\}\}.$$

  Then, the same argument leads to the result that the inequality $\sum_{v\in H} x_v \leq 2q+1$ induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[H])$.

(ii) To conclude the proof, we show that (4.7) is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$ if $q = r = 1$, i.e., $|H| = 4$. By (i), $\sum_{i\in H} x_i \leq 2$ defines a facet of $\mathscr{P}_{\mathcal{IUC}}(G[H])$. Let $\sum_{i\in H} x_i + \sum_{v\in V\setminus H} a_v x_v \leq 2$ be the inequality obtained from an arbitrary sequential lifting of $\sum_{i\in H} x_i \leq 2$ (to $\mathbb{R}^n$). Observe that, for every vertex $v \in V\setminus H$, there always exists two vertices $i, j \in H$ such that $\{i, j, v\} \notin \Lambda(G)$, thus, $\max\{\sum_{i\in H} x_i \mid x \in \mathscr{P}_{\mathcal{IUC}}(G)$ and $x_v = 1\} = 2$. This implies $a_v = 0, \forall v \in V\setminus H$, hence $\sum_{i\in H} x_i \leq 2$ induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G)$. □

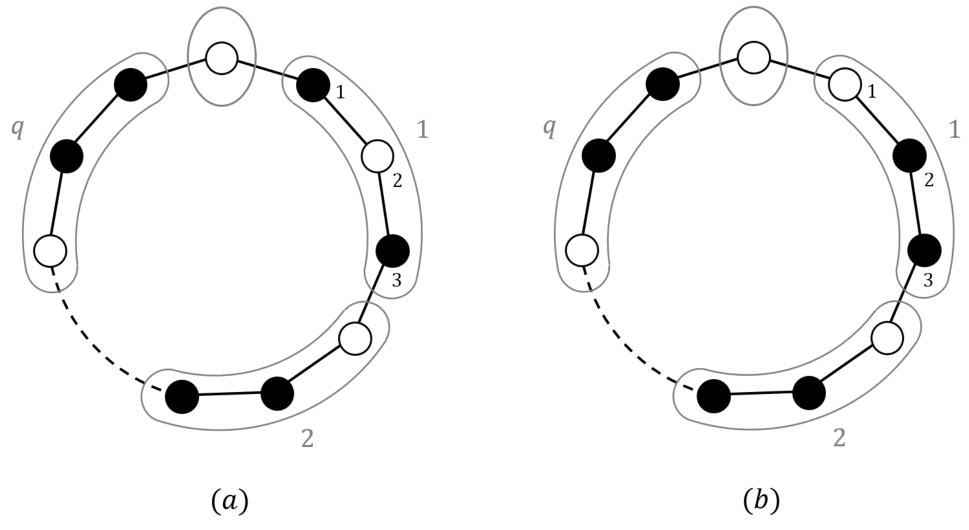

Figure 4.1: Subgraph induced by a hole H, $|H| = 3q + 1$.

It is easy to see the similarity between (4.7) and the *odd-hole* inequality for the vertex packing (independent set in the sense of graph theory) polytope associated with G. Padberg [108] proved that, given a hole H in G with an odd number of vertices, the inequality $\sum_{i \in H} x_i \leq \frac{1}{2}(|H| - 1)$ is facet-defining for the vertex packing polytope associated with $G[H]$. Later, Nemhauser and Trotter [106] noted that the odd-hole inequality is a special case of a more general result concerning independence systems, stated as follows:

Theorem 10. [106] *Suppose α_0 is the cardinality of a maximum independent set in an independence system $\mathscr{S} = (S, \mathcal{S})$, and $\mathscr{S}$ contains $n = |S|$ maximum independent sets $I_1, I_2, \ldots, I_n$ with corresponding affinely independent (incidence) vectors $x^1, x^2, \ldots, x^n$. Then $\sum_{i \in S} x_i \leq \alpha_0$ is facet-defining for the polytope associated with $\mathscr{S}$, i.e., the convex hull of the incidence vectors of all members of $\mathcal{S}$ in $\mathbb{R}^n$.*

Similar to the odd-hole inequality for the vertex packing polytope, Theorem 9(i) spots a facet-inducing inequality implied by Theorem 10 for the independence system defined by the IUC property on $G[H]$. Note that Theorem 10 itself is of little practical value as it provides no means to identify the independence systems satisfying such properties.

Furthermore, Theorems 8 and 9(ii) correspond to *maximal clique* inequality of independence systems. An independence system $(S, \mathcal{S})$ is called k-regular if every circuit of it is of cardinality k. Referring to a k-regular independence system $(S, \mathcal{S})$, $C \subseteq S$ is called a *clique* if $|C| \geq k$ and all $\binom{|C|}{k}$ subsets of C are circuits of $(S, \mathcal{S})$. Clearly, this definition coincides with the graph-theoretic definition of a clique. Nemhauser and Trotter [106] also proved that the inequality $\sum_{i \in C} x_i \leq k-1$ is facet-inducing for the polytope associated with a k-regular independence system $(S, \mathcal{S})$ if $C \subseteq S$ is a maximal (inclusion-wise) clique. A special case of this result for the vertex packing polytope associated with a graph G had been originally shown by Padberg [108]. Let $\mathcal{U}$ denote the set of all IUCs in G, and $(V, \mathcal{U})$ be the corresponding independence system. Patently, every $\{i, j, k\} \in \Lambda(G)$ is a clique in $(V, \mathcal{U})$, and such a clique is maximal only if it is not contained in a clique of cardinality 4. It is not hard to see that a clique of size 4 in $(V, \mathcal{U})$ must be a hole in G. Therefore, Theorem 8 corresponds to maximality of $\{i, j, k\} \in \Lambda(G)$ as a clique of the corresponding independence system. Besides, by the proof of Theorem 9(ii), it became clear that $(V, \mathcal{U})$ cannot have a clique of cardinality greater than 4 because, for every 4-hole H and every vertex $v \in V \backslash H$, there always exist two vertices $i, j \in H$ such that $\{i, j, v\} \notin \Lambda(G)$. Thus, a hole of cardinality 4 in G is actually a maximum clique in $(V, \mathcal{U})$, and Theorem 9(ii) corresponds to its maximality. This result is of a special value for the maximum IUC problem as all facets of this type can be identified in polynomial time. Recall that the vertex packing polytope associated with G may possess an exponential number of such facets.

Given this result, we extend the concept of *fractional clique/independence number* of a graph G to the independence system $(V, \mathcal{U})$. The fractional clique number of G is the maximum sum of nonnegative weights that can be assigned to its vertices such that the total weight of every independent set in the graph is at most 1. Naturally, the fractional clique number of G is equivalent to the fractional independence number of its complement graph, where the total weight of every clique is at most 1. We define the *fractional IUC number* of a graph G as the maximum sum of nonnegative (and no more than 1) weights that can be assigned to its vertices such that the total weight of every clique in $(V, \mathcal{U})$, i.e., the vertex set of every open triangle and 4-hole in G, is at most 2. Clearly, the fractional IUC number of G provides an upper bound on its IUC number, which is computable in polynomial time as opposed to the fractional clique and independence numbers of a graph which are NP-hard to compute [75].

The vertex packing polytope associated with a cycle graph is characterized by the set of maximal clique inequalities, i.e., $x_i + x_j \leq 1,\ \forall \{i,j\} \in E$, the variable (lower) bounds, and the Padberg's odd-hole inequality if the number of vertices is odd. However, the IUC polytope associated with a cycle graph has more facets than those defined by these families. As a case in point, consider a cycle graph on $n = 12$ vertices, and observe that the point given by $x_i = 1,\ \forall i \in \{1,5,9\}$, and $x_i = \frac{1}{2},\ \forall i \in V \backslash \{1,5,9\}$, is an extreme point of the polytope defined by the corresponding set of OT inequalities and the variable bounds. In fact, a fractional extreme point of a similar structure—generated by OT inequalities and variable bounds—exists for every cycle graph on $n = 3q_1 + 4q_2$ vertices, for some nonnegative integers q_1 and q_2, if q_2 is odd. In such a cycle graph, fixing $q_1 + q_2$ variables whose vertices are located at distances 3 or 4 from each other at 1 turns the set of OT inequalities into the set of vertex packing inequalities associated with a conflict cycle graph with an odd number of vertices; this leads to extremity of a point with half integral values on the remaining $n - (q_1 + q_2)$ variables. It is easy to check that, unless $q_2 = 1$, this fractional point always satisfies the hole inequality (4.7).

Given the definition of IUC, it is also natural to consider the correspondence between the IUC polytope and the clique polytope associated with a graph G, i.e., the convex hull of the incidence vectors of all cliques in G. Patently, the odd-hole inequality for the vertex packing polytope translates to the *odd-anti-hole* inequality for the clique polytope associated with G. An *anti-hole* is a subset of vertices $A \subseteq V,\ |A| \geq 4$, such that the (edge) complement of the corresponding induced subgraph is a chordless cycle, i.e., A is a hole in the complement graph of G. The following theorem states that, under a slightly stronger condition, the same inequality is valid for the IUC polytope associated with G and possesses the same facial property on the polytope associated with the corresponding induced subgraph.

Theorem 11. (Anti-hole Inequality) *Let $A \subseteq V,\ |A| \geq 6$, be an anti-hole in $G = (V,E)$. Then,* *the inequality*

$$\sum_{i \in A} x_i \leq \left\lfloor \frac{|A|}{2} \right\rfloor \tag{4.8}$$

is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$. Furthermore, it induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[A])$ if and only if $|A|$ is odd.

Proof. Consider a labeling of the vertices of an anti-hole A, $|A| \geq 6$, such that $\{i, i+1\}, \forall i \in \{1, \dots, |A|-1\}$, and $\{|A|, 1\}$ identify the pairs of non-adjacent vertices in $G[A]$. For a fixed sequence of labels, note that $\{1, 2, i, i+1\}, \forall i \in \{4, \dots, |A|-2\}$, is a hole of cardinality 4, thus the inequality $x_1 + x_2 + x_i + x_{i+1} \leq 2, \forall i \in \{4, \dots, |A|-2\}$, is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$. Consider the summation of those $|A|-5$ valid inequalities. Observe that each vertex $j \in \{1, 2\}$ appears $|A|-5$ times, $j \in \{4, |A|-1\}$ once, and $j \in \{5, \dots, |A|-2\}$ twice, in the left-hand side of the resultant inequality. These partitions are illustrated by black, hatched, and gray vertices in Figure 4.2, respectively. The white vertices in Figure 4.2 are those that are not present in this inequality. Since the starting vertex of the labeling sequence was arbitrary, $|A|$ valid inequalities of this type may be written for $\mathscr{P}_{\mathcal{IUC}}(G)$. Considering all those inequalities, every vertex will appear exactly twice in a black position, twice in a white, twice in a hatched, and $|A|-6$ times in a gray position. As a result, the summation of all 4-hole inequalities associated with $G[A]$ results in

$$\Big(2(|A|-5) + 2 + 2(|A|-6)\Big) \sum_{i \in A} x_i \leq 2|A|(|A|-5).$$

Therefore, $\sum_{i \in A} x_i \leq \frac{1}{2}|A|$ is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$, which further implies validity of (4.8) as the Gomory-Chvátal cut corresponding to this inequality. Clearly, when $|A|$ is even, (4.8) is a linear combination of the 4-hole inequalities associated with $G[A]$, hence not facet-defining for the corresponding polytope. Thus, it remains to show that it induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[A])$ if $|A|$ is odd.

Given an odd anti-hole A, consider $\mathcal{F} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G[A]) \mid \sum_{v \in A} x_v = \frac{1}{2}(|A|-1)\}$. Associated with a fixed labeling of A as described above, let x_C be the incidence vector of the set of even vertices, i.e., $C = \{2, 4, \dots, |A|-1\}$. Observe that C is a clique in $G[A]$ and $x_C \in \mathcal{F}$, which implies $\mathcal{F}$ is a proper face of $\mathscr{P}_{\mathcal{IUC}}(G[A])$. Suppose that $\mathcal{F}$ is contained in a supporting hyperplane

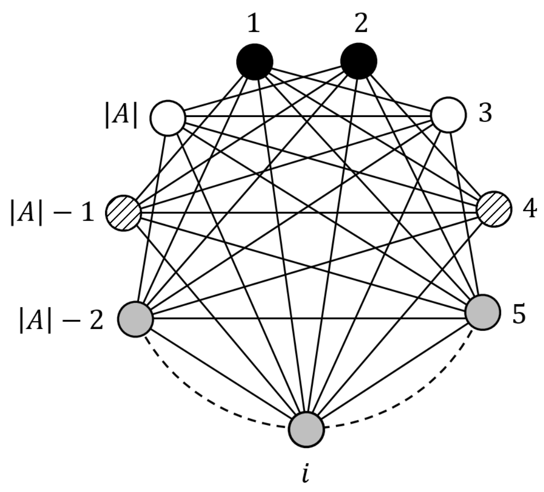

Figure 4.2: Subgraph induced by an anti-hole.

$\mathscr{H} : \sum_{i \in A} a_i x_i = b$ of $\mathscr{P}_{\mathcal{IUC}}(G[A])$. Let $C' = (C \cup \{1\}) \backslash \{2\}$ and note that C' is also a clique, hence $x_{C'} \in \mathcal{F}$. Then, $x_C, x_{C'} \in \mathscr{H}$ implies that $a_1 = a_2$. Since the start vertex of the labeling was arbitrary, this results holds for all pairs of vertices with consecutive labels. That is, $a_i = a, \forall i \in A$, and $b = \frac{1}{2}a(|A| - 1)$. The proof is complete noting that the interior of $\mathscr{P}_{\mathcal{IUC}}(G[A])$ is nonempty, thus $a \neq 0$. $\square$

Theorem 11 is closely related to a basic property of IUCs. Ertem et al. [81] showed that, in an arbitrary graph, a (maximal) clique is a maximal IUC if and only if it is a dominating set. In an anti-cycle graph $G = (A, E)$, each maximal clique is maximum, and in case $|A| \geq 6$, it is also a maximum IUC by Theorem 11. It is easy to verify that the anti-cycle graphs on 4 and 5 vertices are the only ones in which a maximal clique is not a dominating set.

In the subsequent sections, we will show that the IUC polytope associated with an anti-cycle graph has many other facets than those induced by the 4-hole inequalities and (4.8), due to the *fan* substructures that appear in this graph. We postpone our discussion in this regard to Section 4.3.3.

Finally, it should be mentioned that the odd-hole and odd-anti-hole inequalities for the vertex packing polytope have been generalized by Trotter [109] via introducing a general class of facet-producing subgraphs, called *webs*, that subsumes cycle and anti-cycle graphs as special cases.

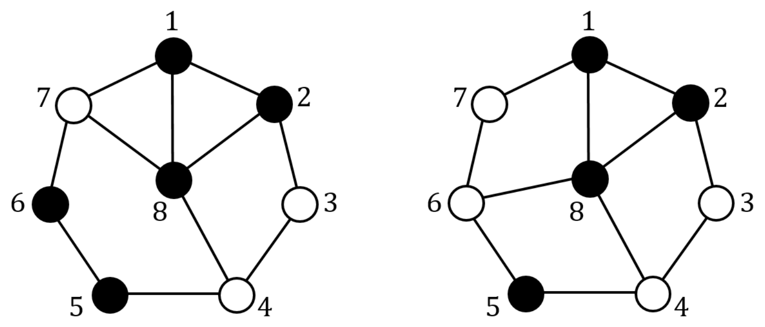

Figure 4.3: Example graphs for lifting the hole inequality.

Given the connection between the vertex packing and IUC polytopes, webs may lead to further facet-inducing inequalities for the IUC polytope as well. We leave this topic for future research and will not pursue it in this paper.

Lifting hole and anti-hole inequalities

Let $G[H]$, $|H| \geq 5$, be an induced chordless cycle in $G = (V, E)$, where $|H| = 3q + r$ and $r \neq 0$. Then by Theorem 9, inequality (4.7) induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[H])$, but it is not necessarily facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$. Consider lifting a variable $x_v,\ v \in V\backslash H$, into (4.7) to generate a facet-defining inequality for $\mathscr{P}_{\mathcal{IUC}}(G[H \cup \{v\}])$ of the form $a_v x_v + \sum_{i \in H} x_i \leq 2q + \left\lfloor \frac{2r}{3} \right\rfloor$. The value of a_v in such an inequality depends on cardinality of $H \cap N(v)$ as well as distances (lengths of the shortest paths) among the vertices of $N(v)$ in $G[H]$. As a case in point, consider the graphs of Figure 4.3, and lifting x_8 into the inequality $\sum_{i=1}^{7} x_i \leq 4$ corresponding to $H = \{1, \ldots, 7\}$. In both graphs, $|H \cap N(8)| = 4$ and the difference is due to the distance between the vertices 4 and 7 (equivalently, 1 and 7) in the left-hand-side graph, and the vertices 4 and 6 (equivalently, 1 and 6) in the right-hand-side graph. The sets of black vertices illustrate the maximum size IUCs containing vertex 8 in these graphs. Clearly, the coefficient of x_8 vanishes in lifting $\sum_{i=1}^{7} x_i \leq 4$ for the left-hand-side graph, indicating that this inequality is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G[H \cup \{8\}])$. On the other hand, for the IUC polytope associated with the right-hand-side graph, this inequality is dominated by the lifted (facet-defining) inequality $x_8 + \sum_{i=1}^{7} x_i \leq 4$.

Obviously, the structure of $N(v)$ in $G[H]$ becomes irrelevant at extreme values of $|H \cap N(v)|$.

This result can be slightly strengthened by showing that the lengths of the paths among the neighbors of v in $G[H]$ are redundant when $|H \cap N(v)|$ is restricted to 3 and 2 for $r = 1$ and $r = 2$, respectively. This leads to the following theorem concerning the lifting procedure of the hole inequality to higher-dimensional spaces.

Theorem 12. *Consider the hole inequality* (4.7) *where $r \neq 0$. In every sequential lifting of* (4.7)*, the coefficient of a variable x_v, $v \in V \backslash H$, vanishes under the following conditions:* (i) $|H \cap N(v)| \leq 3$ *if* $r = 1$, (ii) $|H \cap N(v)| \leq 2$ *if* $r = 2$.

Proof. In an arbitrary sequential lifting of the hole inequality (4.7), even if some variables not-satisfying the theorem conditions have already been lifted into (4.7) before x_v, existence of $2q + \left\lfloor \frac{2r}{3} \right\rfloor$ vertices in H forming an IUC with v is sufficient for the coefficient of x_v to vanish. Thus, we just need to establish the existence of these vertices under the theorem conditions. Such structures for all possible distributions of neighbors of v in $G[H]$ are presented below.

Consider a sequential partitioning of H as described in the proof of Theorem 9. In a clockwise walk, let v_i^p denote the i-th vertex in the p-th partition, where $i \in \{1, 2, 3\}$, $p \in \{1, \ldots, q+1\}$, and $p = q + 1$ is the partition with less than 3 vertices. The cases where $|H \cap N(v)| \leq 1$ are trivial, so we start with $|H \cap N(v)| = 2$. Let u and w be the neighbors of v in H, i.e., $H \cap N(v) = \{u, w\}$, and $d = d_{G[H]}(u, w)$ denote the distance (length of the shorter path) between u and w in $G[H]$. Without loss of generality, we assume $u = v_1^1$ and the shorter path between u and w in $G[H]$ is due to a clockwise walk from u toward w. We treat $d \in \{1, 2\}$ separately as those are the only cases where u and w are located in the same partition. For each case, we present an IUC of size $2q + \left\lfloor \frac{2r}{3} \right\rfloor + 1$ in $G[H \cup \{v\}]$. In the following presentation, each set of vertices is a clique, and j denotes the partition that contains w.

$$d = 1,\ r = 1: \quad \{v, u = v_1^1, w = v_2^1\};\ \{v_1^p, v_2^p\}, \forall p \in \{2, \ldots, q\}$$

$$d = 1,\ r = 2: \quad \{v, u = v_1^1, w = v_2^1\};\ \{v_1^p, v_2^p\}, \forall p \in \{2, \ldots, q\};\ \{v_1^{q+1}\}$$

$$d = 2,\ r = 1: \quad \{v\};\ \{v_2^1\};\ \{v_1^p, v_2^p\}, \forall p \in \{2, \ldots, q\};\ \{v_1^{q+1}\}$$

$$d = 2,\ r = 2: \quad \{v\};\ \{v_2^1\};\ \{v_1^p, v_2^p\}, \forall p \in \{2, \ldots, q\};\ \{v_1^{q+1}, v_2^{q+1}\}$$

$$\begin{aligned}
d \equiv 0 \text{ (mod 3)},\ r = 1 :&\quad \{v\};\ \{v_2^p, v_3^p\}, \forall p \in \{1, \ldots, q\}\\
d \equiv 0 \text{ (mod 3)},\ r = 2 :&\quad \{v\};\ \{v_2^p, v_3^p\}, \forall p \in \{1, \ldots, q\};\ \{v_2^{q+1}\}\\
d \equiv 1 \text{ (mod 3)},\ r = 1 :&\quad \{v, w = v_2^j\};\ \{v_2^p, v_3^p\}, \forall p \in \{1, \ldots, j-1\};\\
&\quad \{v_1^p, v_2^p\}, \forall p \in \{j+1, \ldots, q\};\ \{v_1^{q+1}\}\\
d \equiv 1 \text{ (mod 3)},\ r = 2 :&\quad \{v, w = v_2^j\};\ \{v_2^p, v_3^p\}, \forall p \in \{1, \ldots, j-1\};\\
&\quad \{v_1^p, v_2^p\}, \forall p \in \{j+1, \ldots, q\};\ \{v_1^{q+1}, v_2^{q+1}\}\\
d \equiv 2 \text{ (mod 3)},\ r = 1 :&\quad \{v\};\ \{v_2^p, v_3^p\}, \forall p \in \{1, \ldots, j-1\};\ \{v_2^j\};\\
&\quad \{v_1^p, v_2^p\}, \forall p \in \{j+1, \ldots, q\};\ \{v_1^{q+1}\}\\
d \equiv 2 \text{ (mod 3)},\ r = 2 :&\quad \{v\};\ \{v_2^p, v_3^p\}, \forall p \in \{1, \ldots, j-1\};\ \{v_2^j\};\\
&\quad \{v_1^p, v_2^p\}, \forall p \in \{j+1, \ldots, q\};\ \{v_1^{q+1}, v_2^{q+1}\}
\end{aligned}$$

Next, we consider $|H \cap N(v)| = 3$ and $r = 1$. Let $H \cap N(v) = \{u, w, z\}$. As before, suppose that $u = v_1^1$, and in a clockwise walk starting from u, first w is reached in $d_1 = d_{G[H]}(u, w)$ steps, and then z is reached in $d_2 = d_{G[H]}(w, z)$ steps from w. We also assume $d_1 \leq d_2$. In the following presentation, j and k denote the partitions containing w and z, respectively. First, we consider the cases where u and w are located in the same partition, i.e., $d_1 \in \{1, 2\}$. Note that u, w and z are placed in one partition only if $d_1 = d_2 = 1$.

$$\begin{aligned}
d_1 = 1,\ d_2 = 1 :&\quad \{v, u = v_1^1, w = v_2^1\};\ \{v_1^p, v_2^p\}, \forall p \in \{2, \ldots, q\}\\
d_1 = 1,\ d_2 \equiv 0 \text{ (mod 3)} :&\quad \{v, u = v_1^1, w = v_2^1\};\ \{v_1^2\};\\
&\quad \{v_3^p, v_1^{p+1}\}, \forall p \in \{2, \ldots, q-1\};\ \{v_3^q\}\\
d_1 = 1,\ d_2 \equiv 1 \text{ (mod 3)} :&\quad \{v, u = v_1^1, w = v_2^1\};\ \{v_1^p, v_2^p\}, \forall p \in \{2, \ldots, q\}\\
d_1 = 1,\ d_2 \equiv 2 \text{ (mod 3)} :&\quad \{v, u = v_1^1, w = v_2^1\};\ \{v_2^p, v_3^p\}, \forall p \in \{2, \ldots, q\}\\
d_1 = 2,\ d_2 \equiv 0 \text{ (mod 3)} :&\quad \{v\};\ \{v_2^1\};\ \{v_1^p, v_2^p\}, \forall p \in \{2, \ldots, q\};\ \{v_1^{q+1}\}\\
d_1 = 2,\ d_2 \equiv 1 \text{ (mod 3)} :&\quad \{v, z = v_1^k\};\ \{v_2^1\};\ \{v_1^p, v_2^p\}, \forall p \in \{2, \ldots, k-1\};\\
&\quad \{v_3^p, v_1^{p+1}\}, \forall p \in \{k, \ldots, q\}
\end{aligned}$$

$$d_1 = 2,\ d_2 \equiv 2 \text{ (mod 3)}: \quad \{v\};\ \{v_2^1\};\ \{v_1^2\};\ \{v_3^p, v_1^{p+1}\}, \forall p \in \{2,\dots,q\}$$

Finally, the following IUCs correspond to the cases where u, w and z are all located in separate partitions.

$$\begin{aligned}
d_1 \equiv 0 \text{ (mod 3)},\ d_2 \equiv 0 \text{ (mod 3)}: \quad & \{v\};\ \{v_2^p, v_3^p\}, \forall p \in \{1,\dots,q\} \\
d_1 \equiv 0 \text{ (mod 3)},\ d_2 \equiv 1 \text{ (mod 3)}: \quad & \{v, z = v_2^k\};\ \{v_2^p, v_3^p\}, \forall p \in \{1,\dots,k-1\}; \\
& \{v_1^p, v_2^p\}, \forall p \in \{k+1,\dots,q\};\ \{v_1^{q+1}\} \\
d_1 \equiv 0 \text{ (mod 3)},\ d_2 \equiv 2 \text{ (mod 3)}: \quad & \{v\};\ \{v_2^p, v_3^p\}, \forall p \in \{1,\dots,k-1\};\ \{v_2^k\}; \\
& \{v_1^p, v_2^p\}, \forall p \in \{k+1,\dots,q\};\ \{v_1^{q+1}\} \\
d_1 \equiv 1 \text{ (mod 3)},\ d_2 \equiv 0 \text{ (mod 3)}: \quad & \{v, w = v_2^j\};\ \{v_2^p, v_3^p\}, \forall p \in \{1,\dots,j-1\}; \\
& \{v_1^{j+1}\};\ \{v_3^p, v_1^{p+1}\}, \forall p \in \{j+1,\dots,q\} \\
d_1 \equiv 1 \text{ (mod 3)},\ d_2 \equiv 1 \text{ (mod 3)}: \quad & \{v, w = v_2^j\};\ \{v_2^p, v_3^p\}, \forall p \in \{1,\dots,j-1\}; \\
& \{v_1^p, v_2^p\}, \forall p \in \{j+1,\dots,q\};\ \{v_1^{q+1}\} \\
d_1 \equiv 1 \text{ (mod 3)},\ d_2 \equiv 2 \text{ (mod 3)}: \quad & \{v, u = v_1^1\};\ \{v_3^p, v_1^{p+1}\}, \forall p \in \{1,\dots,j-1\}; \\
& \{v_3^p, v_1^{p+1}\}, \forall p \in \{j,\dots,k-2\};\ \{v_3^{k-1}\}; \\
& \{v_2^p, v_3^p\}, \forall p \in \{k,\dots,q\} \\
d_1 \equiv 2 \text{ (mod 3)},\ d_2 \equiv 0 \text{ (mod 3)}: \quad & \{v\};\ \{v_2^p, v_3^p\}, \forall p \in \{1,\dots,j-1\}; \\
& \{v_2^j\};\ \{v_1^p, v_2^p\}, \forall p \in \{j+1,\dots,q\};\ \{v_1^{q+1}\} \\
d_1 \equiv 2 \text{ (mod 3)},\ d_2 \equiv 1 \text{ (mod 3)}: \quad & \{v, z = v_1^k\};\ \{v_2^p, v_3^p\}, \forall p \in \{1,\dots,j-1\}; \\
& \{v_2^j\};\ \{v_1^p, v_2^p\}, \forall p \in \{j+1,\dots,k-1\}; \\
& \{v_3^p, v_1^{p+1}\}, \forall p \in \{k,\dots,q\} \\
d_1 \equiv 2 \text{ (mod 3)},\ d_2 \equiv 2 \text{ (mod 3)}: \quad & \{v\};\ \{v_2^p, v_3^p\}, \forall p \in \{1,\dots,j-1\}; \\
& \{v_2^j\};\ \{v_1^{j+1}\};\ \{v_3^p, v_1^{p+1}\}, \forall p \in \{j+1,\dots,q\}
\end{aligned}$$

The proof is complete. $\square$

In Section 4.3.3, we will discuss implication of this theorem about the facial structure of the IUC polytope associated with a *defective wheel* graph.

We conclude this section by stating a similar result concerning the anti-hole inequality. By construction of a maximum IUC in an anti-cycle graph, presented it the proof of Theorem 11, it is easy to verify that the coefficient of a variable $x_v, v \in V \backslash A$, vanishes in lifting the odd-anti-hole inequality (4.8) to higher-dimensional spaces if $|A \cap N(v)| \in \{0, 1, |A|-1, |A|\}$. The following theorem mainly asserts that distribution of the neighbors of v in $G[A]$ is not relevant in lifting x_v into (4.8) when $|A \cap N(v)| \in \{2, |A|-2\}$, and the corresponding coefficient vanishes under this condition as well.

Theorem 13. *Consider the odd-anti-hole inequality* (4.8)*. In every sequential lifting of* (4.8)*, the coefficient of a variable $x_v, v \in V \backslash A$, vanishes if $|A \cap N(v)| \leq 2$ or $|A \cap N(v)| \geq |A| - 2$.*

Proof. We show that if $|A \cap N(v)| \leq 2$, then $G[A]$ always contains a clique C of cardinality $|C| = \frac{1}{2}(|A| - 1)$ such that $C \cap N(v) = \emptyset$, and if $|A \cap N(v)| \geq |A| - 2$, then $G[A \cup \{v\}]$ has a clique of size $\frac{1}{2}(|A| - 1) + 1$. The cases $|A \cap N(v)| \leq 1$ and $|A \cap N(v)| \geq |A| - 1$ are trivial. Let $|A \cap N(v)| = 2$, and suppose u and w are the neighbors of v in A. Consider a clockwise labeling of A starting from u as described in the proof of Theorem 11. Observe that, if the label of w is odd, then the set of vertices with even labels form a clique of size $\frac{1}{2}(|A| - 1)$, neither of which is connected to v. On the other hand, if the label of w is even, it will become odd in a counterclockwise labeling of A starting from u, as A itself is of odd cardinality. The clique of interest is then given by the even-labeled vertices in the counterclockwise labeling. The proof for the case $|A \cap N(v)| = |A| - 2$ is identical except that u and w are defined to be the vertices in A not adjacent to v. The proof is complete. $\square$

4.3.2 Star and double-star

A *star* graph on n vertices is a tree composed of a central vertex, called *hub*, connected to $n - 1$ leaves. We use the notation $S = \{h\} \cup I$ to refer to the vertex set of a star graph. Here, h is the hub vertex and I denotes the set of leaves, remarking that they form an independent set in the graph.

Theorem 14. (Star Inequality) *Let $S = \{h\} \cup I$ be a subset of vertices inducing a star graph in $G = (V, E)$. Then, the inequality*

$$\sum_{i \in I} x_i + (|I| - 1)x_h \leq |I| \tag{4.9}$$

is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$. Furthermore, (i) *it induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[S])$, and* (ii) *it is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$ if and only if there does not exist a vertex $v \in V \backslash N[h]$ such that $S' = \{v\} \cup I$ induces another star graph in G.*

Proof. Validity of (4.9) is evident. Let $\hat{x}$ be the incidence vector of an arbitrary IUC in G. If $\hat{x}_h = 0$, then (4.9) is trivially valid. Otherwise, at most one vertex from I may belong to this IUC, since $\{h, i, j\} \in \Lambda(G), \forall i, j \in I$.

(i) To prove that (4.9) is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G[S])$, it is sufficient to point out that $x_I = \sum_{i \in I} e_i$ and $x_{\{h,i\}} = e_h + e_i, \forall i \in I$, are $|I| + 1$ affinely independent points belonging to $\mathcal{F} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G[S]) \mid \sum_{i \in I} x_i + (|I| - 1)x_h = |I|\}$.

(ii) To see the necessity of this condition, note that if there exist two star subgraphs in G induced by $S = \{h\} \cup I$ and $S' = \{v\} \cup I$, for some $v \in V \backslash N[h]$, then the star inequality corresponding to S is dominated by $\sum_{i \in I} x_i + (|I| - 1)x_h + x_v \leq |I|$, which is obtained from lifting x_v into (4.9).

To prove sufficiency, assume that there does not exist a vertex $v \in V \backslash N[h]$ such that $S' = \{v\} \cup I$ induces a star graph in G. Let $\sum_{v \in V \backslash S} a_v x_v + \sum_{i \in I} x_i + (|I| - 1)x_h \leq |I|$ be the inequality obtained from an arbitrary sequential lifting of (4.9). We need to show that $a_v = 0, \forall v \in V \backslash S$. In this regard, consider the following cases:

- $v \in N(h)$

  If v is also adjacent to a vertex $i \in I$, then $\{h, i, v\}$ is a clique. Otherwise, $\{v\} \cup I$ is an independent set in G.

- $v \notin N(h)$

  Since $\{v\} \cup I$ does not induce a star subgraph, there exists a vertex $i \in I$ not adjacent to v. Thus, $G[\{h, i, v\}]$ is the union of an isolated vertex v and an edge $\{h, i\}$.

Therefore, there always exists an IUC in $G[S \cup \{v\}]$ containing v whose incidence vector satisfies $\sum_{i\in I} x_i + (|I|-1)x_h = |I|$. This results implies that $a_v = 0, \forall v \in V\backslash S$, hence (4.9) is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$ under the theorem's condition. □

It is interesting to note that no further condition, such as maximality of I in the neighborhood of h, is required for (4.9) to be facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$. In fact, in absence of a vertex $v \in V\backslash N[h]$ described in the theorem statement, every $I' \subseteq I, |I'| \geq 2$, corresponds to a distinct facet of $\mathscr{P}_{\mathcal{IUC}}(G)$ defined by (4.9). In particular, every minimal (inclusion-wise) of such subsets corresponds to a facet-defining OT inequality. Recall that an OT inequality corresponding to $\{h, i, j\}$, for some $\{i, j\} \subseteq I$, is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$ if and only if $\{h, i, j, v\}, \forall v \in V\backslash\{h, i, j\}$, is not a 4-hole, or equivalently, $\{v, i, j\}$ does not induce a star subgraph.

The proof of Theorem 14 also declares that if two star subgraphs with the same set of leaves exist in G, then the family of star inequalities (4.9) corresponding to each one of them can be lifted to a higher-dimensional space to generate facet-inducing inequalities for the corresponding *double-star* (complete bipartite $K_{|I|,2}$) subgraph. The following theorem shows that these double-star inequalities are actually facet-inducing for $\mathscr{P}_{\mathcal{IUC}}(G)$, which also generalizes the facial property of the 4-hole inequality.

Theorem 15. (Double-star Inequality) *Let $S_h = \{h\} \cup I$ and $S_u = \{u\} \cup I$ be two sets of vertices inducing star subgraphs in G, where h and $u \in V\backslash N[h]$ denote two (non-adjacent) hub vertices, and I is the joint set of leaves. Then, the inequalities*

$$\sum_{i\in I} x_i + (|I|-1)x_h + x_u \leq |I| \quad \textit{and} \quad \sum_{i\in I} x_i + x_h + (|I|-1)x_u \leq |I| \tag{4.10}$$

induce facets of $\mathscr{P}_{\mathcal{IUC}}(G)$.

Proof. We present the proof for $\sum_{i\in I} x_i + (|I|-1)x_h + x_u \leq |I|$. The other follows from the symmetry of this structure. To this end, we just need to show that, for every $v \in V\backslash(\{h, u\} \cup I)$, there always exists an IUC containing v in the subgraph $G[\{h, u, v\} \cup I]$ whose incidence vector satisfies $\sum_{i\in I} x_i + (|I|-1)x_h + x_u = |I|$. This result will be sufficient to indicate that the coefficient of a variable $x_v, \forall v \in V\backslash(\{h,u\}\cup I)$ vanishes in every sequential lifting of $\sum_{i\in I} x_i+(|I|-1)x_h+ x_u \leq |I|$, proving that this inequality is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$. The existence of such IUCs when $v \in N(h)$ is established by the proof of Theorem 14. On the other hand, if $v \notin N(h)$, then $\{h,u,v\}$ is an IUC, that is either an independent set or the union of an isolated vertex with two adjacent vertices. This completes the proof. □

It is easy to see that if $G=(S,E),\ S=\{h\}\cup I$, is a star graph, then a subset of vertices in G is an IUC if and only if it is a *co-1-defective clique*. An s-defective clique is a subset of vertices C such that the corresponding induced subgraph contains at least $\binom{|C|}{2}-s$ edges [77]. Correspondingly, a co-s-defective clique is a subset of vertices whose induced subgraph has at most s edges. Sherali and Smith [110] have shown that the family of inequalities defined by (4.9) for all $I' \subseteq I, |I'| \geq 2$, along with the variable bounds, is sufficient to describe the co-1-defective clique polytope associated with a star graph. Given the equivalence of these two structures in a star graph, this result also holds for the IUC polytope associated with G, which is presented through the following theorem.

Theorem 16. *The family of inequalities* (4.9) *together with the variable bounds provide a complete description for the IUC polytope associated with a star graph $G=(S,E)$.*

Proof. See Proposition 3 in [110] and its proof. □

In [110], the authors use the term *generalized vertex packing* to refer to co-s-defective clique. We have used the latter term to avoid confusion with the concept of *generalized independent set*, which has also been the subject of some recent studies [111, 112].

We extend this result to the IUC polytope associated with a (general) tree. Every tree can be seen as a collection of connected stars, whose hub vertices are the internal (branch) nodes of the tree. The following theorem declares that the family of star inequalities (4.9) generated by the star subgraphs of a tree, together with the variable bounds, is sufficient to describe the IUC polytope associated with this graph.

Theorem 17. *Let $G = (V, E)$ be a tree, and $V_b \subset V$ denote the set of its internal (branch) vertices. Then,*

$$\mathscr{P}_{\mathcal{IUC}}(G) = \{x \in [0,1]^n \mid \sum_{i \in I} x_i + (|I| - 1)x_v \leq |I|,\ \forall v \in V_b,\ \forall I \subseteq N(v), |I| \geq 2\}. \quad (4.11)$$

Proof. We show that every extreme point of the polytope defined by the inequalities in (4.11) is integral. The proof is by induction. The base case is a star graph, for which the result holds by Theorem 16. Patently, every tree $G = (V, E)$ is constructed by appending a leaf vertex $j \in V$ to the subtree $G[V \backslash \{j\}]$. So, suppose that the result holds for every induced subtree of $G[V \backslash \{j\}]$; we show that it also holds for G. Let h denote the parent of j, i.e., the internal vertex adjacent to j, in G, and without loss of generality, assume that G is rooted at h. Furthermore, let $N(h) \backslash \{j\} = \{r_k,\ \forall k \in \{1, \dots, |N(h)| - 1\}\}$, and $G^k,\ \forall k \in \{1, \dots, |N(h)| - 1\}\}$, denote the subtree of G rooted at r_k together with h as a leaf vertex. We present the proof for the case that h is an internal vertex of $G[V \backslash \{j\}]$; the case that h is a leaf vertex in $G[V \backslash \{j\}]$ becomes trivial after presenting this (more general) result, as it corresponds to $k \in \{1\}$.

Let $\hat{x}$ be an arbitrary extreme point of the polytope defined by the inequalities in (4.11), denoted by $\mathscr{P}$. Evidently, if $\hat{x}_j = 0$ or $\hat{x}_h = 0$, the result trivially holds, by the induction hypothesis. Besides, $\hat{x}_j = \hat{x}_h = 1$ implies $\hat{x}_{r_k} = 0,\ \forall k \in \{1, \dots, |N(h)| - 1\}\}$, which brings about the same result. We show that fractionality of $\hat{x}_j$ or $\hat{x}_h$ contradicts the extremity of $\hat{x}$ for each one of the remaining three possibilities:

(i) $\hat{x}_h = 1$ and $\hat{x}_j \in (0,1) \;\Rightarrow\; \hat{x}_{r_k} \neq 1,\ \forall k \in \{1, \dots, |N(h)| - 1\}\}$

Consider the facet $\mathscr{P}_{(x_h=1)} = \{x \in \mathscr{P} \mid x_h = 1\}$. Observe that, $x_h = 1$ reduces the entire set of star inequalities (4.9) associated with the star subgraph $G[\{h\} \cup N(h)]$ to

$$x_j + \sum_{k=1}^{|N(h)|-1} x_{r_k} \leq 1,$$

in the description of $\mathscr{P}_{(x_h=1)}$, and every other inequality in the description of this facet uniquely appears in the description of the polytope $\mathscr{P}^k_{(x_h=1)} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G^k) \mid x_h = 1\}$, for some $k \in \{1, \ldots, |N(h)| - 1\}$, by the induction hypothesis. Note that, if $\hat{x}_{r_k} = 0$, for some $k \in \{1, \ldots, |N(h)| - 1\}$, we may exclude the subtree rooted at r_k from the graph; so, without loss of generality, we assume $\hat{x}_{r_k} \in (0, 1),\ \forall k \in \{1, \ldots, |N(h)| - 1\}$. Let $V^1 = V(G^1)$ denote the vertex set of G^1, and consider the point $\tilde{x} \in \mathbb{R}^{|V^1|}$ given by $\tilde{x}_i = \hat{x}_i,\ \forall i \in V^1$. By the induction hypothesis, $\tilde{x}$ is not an extreme point of $\mathscr{P}^1_{(x_h=1)} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G^1) \mid x_h = 1\}$. Hence, there always exist two distinct points $\tilde{x}^+ \in \mathscr{P}^1_{(x_h=1)}$ and $\tilde{x}^- \in \mathscr{P}^1_{(x_h=1)}$ such that $\tilde{x}^+_{r_1} = \tilde{x}_{r_1} + \epsilon$ and $\tilde{x}^-_{r_1} = \tilde{x}_{r_1} - \epsilon$, for some $\epsilon \geq 0$, and $\tilde{x} = \frac{1}{2}(\tilde{x}^+ + \tilde{x}^-)$. Notice that $\tilde{x}^+$ and $\tilde{x}^-$ are always two distinct points, and $\epsilon = 0$ corresponds to the event that they differ in a coordinate other than x_{r_1}. This immediately implies that $\hat{x}$ can always be written as a convex combination of two distinct points $\hat{x}^+ \in \mathscr{P}_{(x_h=1)}$ and $\hat{x}^- \in \mathscr{P}_{(x_h=1)}$ given by

$$\hat{x}^+_j = \hat{x}_j - \epsilon;\ \ \hat{x}^+_h = 1;\ \ \hat{x}^+_{r_1} = \hat{x}_{r_1} + \epsilon,\ \ \hat{x}^+_i = \tilde{x}^+_i, \forall i \in V^1 \backslash \{h, r_1\};\ \ \hat{x}^+_i = \hat{x}_i, \forall i \in V \backslash (\{j\} \cup V^1),$$
$$\hat{x}^-_j = \hat{x}_j + \epsilon;\ \ \hat{x}^-_h = 1;\ \ \hat{x}^-_{r_1} = \hat{x}_{r_1} - \epsilon,\ \ \hat{x}^-_i = \tilde{x}^-_i, \forall i \in V^1 \backslash \{h, r_1\};\ \ \hat{x}^-_i = \hat{x}_i, \forall i \in V \backslash (\{j\} \cup V^1),$$

which contradicts the extremity of $\hat{x}$.

(ii) $\hat{x}_h \in (0, 1)$ and $\hat{x}_j = 1 \ \Rightarrow\ \hat{x}_{r_k} \neq 1,\ \forall k \in \{1, \ldots, |N(h)| - 1\}$

As before, we assume $\hat{x}_{r_k} \in (0, 1),\ \forall k \in \{1, \ldots, |N(h)| - 1\}$. In this case, the set of star inequalities (4.9) associated with the star subgraph $G[\{h\} \cup N(h)]$ reduces to

$$\sum_{i \in I} x_i + |I| x_h \leq |I|,\ \forall I \subseteq N(h) \backslash \{j\}, |I| \geq 1. \tag{4.12}$$

Notice that every inequality not involving x_j in this set is redundant in the description of $\mathscr{P}_{(x_j=1)} = \{x \in \mathscr{P} \mid x_j = 1\}$, and (4.12) is due to substituting $x_j = 1$ in the inequalities involving this variable. Besides, every other inequality in the description of $\mathscr{P}_{(x_j=1)}$ uniquely appears in the description of $\mathscr{P}_{\mathcal{IUC}}(G^k)$, for some $k \in \{1, \ldots, |N(h)| - 1\}$, by the induction hypothesis. Observe that every two points $\hat{x}^+ \in \mathbb{R}^n$ and $\hat{x}^- \in \mathbb{R}^n$ that satisfy

$$\begin{aligned}\hat{x}_h^+ &= \hat{x}_h + \epsilon;\;\; \hat{x}_{r_k}^+ = \hat{x}_{r_k} - \epsilon, \forall k \in \{1,\dots,|N(h)|-1\},\\ \hat{x}_h^- &= \hat{x}_h - \epsilon;\;\; \hat{x}_{r_k}^- = \hat{x}_{r_k} + \epsilon, \forall k \in \{1,\dots,|N(h)|-1\},\end{aligned} \tag{4.13}$$

for some $\epsilon \geq 0$, will also satisfy all inequalities in (4.12). We aim to show that, under the induction assumptions, $\hat{x}$ can always be written as $\hat{x} = \frac{1}{2}(\hat{x}^+ + \hat{x}^-)$ for two distinct points $\hat{x}^+ \in \mathbb{R}^n$ and $\hat{x}^- \in \mathbb{R}^n$ that, in addition to (4.13) and $\hat{x}_j^+ = \hat{x}_j^- = \hat{x}_j = 1$, satisfy all defining inequalities of $\mathscr{P}_{\mathcal{IUC}}(G^k),\ \forall k \in \{1,\dots,|N(h)|-1\}\}$. Clearly, we just need to present the proof for $\mathscr{P}_{\mathcal{IUC}}(G^1)$, as x_h is the only variable that the defining inequalities of $\mathscr{P}_{\mathcal{IUC}}(G^k),\ \forall k \in \{1,\dots,|N(h)|-1\}\}$, share. To this end, it is sufficient to show that the point $\tilde{x} \in \mathbb{R}^{|V^1|}$ given by $\tilde{x}_i = \hat{x}_i,\ \forall i \in V^1$, is not an extreme point of the polytope

$$\widetilde{\mathscr{P}} = \mathscr{P}_{\mathcal{IUC}}(G^1)\ \cap\ \{x \in \mathbb{R}^{|V^1|} \mid x_{r_1} + x_h = \tilde{x}_{r_1} + \tilde{x}_h\}.$$

By the induction hypothesis, $\tilde{x}$ is not an extreme point of $\mathscr{P}_{\mathcal{IUC}}(G^1)$. Therefore, if it is an extreme point of $\widetilde{\mathscr{P}}$, it must lie on the interior of an edge of $\mathscr{P}_{\mathcal{IUC}}(G^1)$, and can be written as

$$\tilde{x} = \lambda x^{U_1} + (1-\lambda)x^{U_2},\ \lambda \in (0,1),$$

where x^{U_1} and x^{U_2} are the incidence vectors of two distinct IUCs in G^1. Since $\tilde{x}_{r_1}$ and $\tilde{x}_h$ are not integral, x^{U_1} and x^{U_2} must satisfy one of the following conditions:

(a) $x_{r_1}^{U_1} = x_h^{U_2} = 1$ and $x_{r_1}^{U_2} = x_h^{U_1} = 0$

Thus, $\tilde{x}_{r_1} = \lambda$, $\tilde{x}_h = 1-\lambda$, and $\tilde{x}_{r_1} + \tilde{x}_h = 1$. This implies that every star inequality (4.9) involving x_h, associated with the star subgraph $G^1[\{r_1\} \cup N(r_1)]$, is redundant in the description of $\widetilde{\mathscr{P}}$, as it can be written as a linear combination of $x_{r_1} + x_h \leq 1$ and another star inequality, associated with the subgraph $G^1[S],\ S = \{r_1\} \cup (N(r_1)\backslash\{h\})$, or a variable bound. Consider the point $\bar{x} \in \mathbb{R}^{|V^1|-1}$ given by $\bar{x}_i = \tilde{x}_i,\ \forall i \in V^1\backslash\{h\}$.

By the induction hypothesis, $\bar{x}$ can always be written as a convex combination of two distinct points $\bar{x}^+ \in \mathscr{P}_{\mathcal{IUC}}(G^1[V^1\backslash\{h\}])$ and $\bar{x}^- \in \mathscr{P}_{\mathcal{IUC}}(G^1[V^1\backslash\{h\}])$ such that $\bar{x}^+_{r_1} = \bar{x}_{r_1} + \epsilon$ and $\bar{x}^-_{r_1} = \bar{x}_{r_1} - \epsilon$, for some $\epsilon \geq 0$. Since $x_{r_1} + x_h = 1$ is the only essential inequality involving x_h in the description $\widetilde{\mathscr{P}}$, this result immediately implies that $\bar{x}$ can always be written as a convex combination of two distinct points $\tilde{x}^+ \in \widetilde{\mathscr{P}}$ and $\tilde{x}^- \in \widetilde{\mathscr{P}}$, given by

$$\begin{aligned} &\tilde{x}^+_h = \tilde{x}_h - \epsilon;\ \ \tilde{x}^+_{r_1} = \tilde{x}_{r_1} + \epsilon;\ \ \tilde{x}^+_i = \bar{x}^+_i,\ \forall i \in V^1\backslash\{h, r_1\}, \\ &\tilde{x}^-_h = \tilde{x}_h + \epsilon;\ \ \tilde{x}^-_{r_1} = \tilde{x}_{r_1} - \epsilon;\ \ \tilde{x}^-_i = \bar{x}^-_i,\ \forall i \in V^1\backslash\{h, r_1\}, \end{aligned} \tag{4.14}$$

which contradicts the extremity of $\tilde{x}$.

(b) $x^{U_1}_{r_1} = x^{U_1}_h = 1$ and $x^{U_2}_{r_1} = x^{U_2}_h = 0 \ \Rightarrow \ x^{U_1}_i = 0,\ \forall i \in N(r_1)\backslash\{h\}$

Thus, $\tilde{x}_{r_1} = \tilde{x}_h = \lambda$ and $\tilde{x}_i \in \{0, 1-\lambda\},\ \forall i \in N(r_1)\backslash\{h\}$. This implies that no star inequality (4.9) involving x_h, associated with the star subgraph $G^1[\{r_1\} \cup N(r_1)]$, can be binding at $\tilde{x}$, because otherwise the equality

$$\sum_{i\in I} x_i + x_h + |I|x_{r_1} = |I| + 1,\ I \subseteq N(r_1)\backslash\{h\},\ |I| \geq 1,$$

leads to $\lambda = 1$. Similar to (a), this result further implies the existence of two distinct points $\tilde{x}^+ \in \widetilde{\mathscr{P}}$ and $\tilde{x}^- \in \widetilde{\mathscr{P}}$, defined in (4.14) for a small enough ϵ, which contradicts the extremity of $\tilde{x}$.

(iii) $\hat{x}_h \in (0,1)$ and $\hat{x}_j \in (0,1)$

Consider the following possibilities:

(a) $\hat{x}_h + \hat{x}_j > 1 \ \Rightarrow \ \hat{x}_{r_k} \in [0,1),\ \forall k \in \{1, \ldots, |N(h)| - 1\}$

In this case, no star inequality (4.9) associated with the star subgraph $G[S],\ S = \{h\} \cup (N(h)\backslash\{j\})$, can be tight at $\hat{x}$, because otherwise $\hat{x}$ violates a star inequality involving x_j. Therefore, any star inequality (4.9) associated with the star subgraph $G[\{h\} \cup N(h)]$

binding at $\hat{x}$ is of the form

$$\sum_{i \in I} x_i + x_j + |I| x_h \leq |I| + 1, \; \forall I \subseteq N(h) \backslash \{j\}, |I| \geq 1,$$

and every other inequality in the description of $\mathscr{P}$ uniquely appears in the description of $\mathscr{P}_{\mathcal{IUC}}(G^k)$, for some $k \in \{1, \ldots, |N(h)| - 1\}\}$, by the induction hypothesis. The rest of the proof is identical to (ii), except that here we have $\hat{x}_j^+ = \hat{x}_j^- = \hat{x}_j \neq 1$.

(b) $\hat{x}_h + \hat{x}_j = 1$

If $\hat{x}$ is an extreme point of $\mathscr{P}$, it must also be an extreme point of the polytope

$$\widetilde{\mathscr{P}} = \mathscr{P} \, \cap \, \{x \in \mathbb{R}^n \mid x_h + x_j = 1\}.$$

Then, an argument similar to (ii).(a) will lead to a contradiction with the extremity of $\hat{x}$.

(c) $\hat{x}_h + \hat{x}_j < 1$

This implies that no star inequality (4.9) involving x_j, associated with the star subgraph $G[\{h\} \cup N(h)]$, can be binding at $\hat{x}$, because otherwise $\hat{x}$ violates a star inequality (4.9) associated with the star subgraph $G[S]$, $S = \{h\} \cup (N(h) \backslash \{j\})$. Since the other inequalities in the description of $\mathscr{P}$ are the inequalities defining $\mathscr{P}_{\mathcal{IUC}}(G[V \backslash \{j\}])$, by the induction hypothesis, this is a blatant contradiction with the extremity of $\hat{x}$.

The proof is complete. $\square$

Given a graph G, a subset of vertices is called a *k-dependent set* if the degree of every vertex in the corresponding induced subgraph is at most k. It is evident that IUC and 1-dependent set are equivalent in trees. Hence, the foregoing result also holds for the 1-dependent set polytope associated with a tree, which we present through the following corollary. We should also point out that the MAXIMUM k-DEPENDENT SET problem, which is to find a maximum-cardinality k-dependent set in G, is linear-time solvable on trees, for every $k \geq 1$ [113].

Corollary 18. *Let G be a tree. Then, the family of star inequalities* (4.9) *for all star subgraphs of G, together with the variable bounds, is sufficient to describe the 1-dependent set polytope associated with G.*

We also show that the family of inequalities given by (4.10) plays the same role in description of the IUC polytope associated with a complete bipartite graph $G = K_{|I|,|J|}$. The class of complete bipartite graphs is of a special interest because it has been shown by Pyatkin et al. [114] that a complete bipartite graph $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$ on n vertices accommodates the maximum possible number of open triangles among all the graphs with the same number of vertices.

Theorem 19. *Let $G = (V, E)$ be a complete bipartite graph, where the bipartition of vertices is given by $V = I \cup J$. Then, a complete description of the IUC polytope associated with G is given by the family of inequalities* (4.10) *for all double-star subgraphs of G together with the variable bounds.*

Proof. We assume $|I|, |J| \geq 2$ because otherwise G is a star graph. Let $\mathscr{P}$ be the polytope defined by the family of inequalities (4.10) for all double-star subgraphs of G together with the variable bounds:

$$\mathscr{P} = \{x \in \mathbb{R}^n \mid 0 \leq x_v \leq 1, \ \forall v \in V; \tag{4.15a}$$

$$\forall \{j', j''\} \subseteq J, \ \forall I' \subseteq I, |I'| \geq 2:$$

$$\sum_{i \in I'} x_i + (|I'| - 1)x_{j'} + x_{j''} \leq |I'|, \tag{4.15b}$$

$$\sum_{i \in I'} x_i + x_{j'} + (|I'| - 1)x_{j''} \leq |I'|; \tag{4.15c}$$

$$\forall \{i', i''\} \subseteq I, \ \forall J' \subseteq J, |J'| \geq 2:$$

$$\sum_{j \in J'} x_j + (|J'| - 1)x_{i'} + x_{i''} \leq |J'|, \tag{4.15d}$$

$$\sum_{j \in J'} x_j + x_{i'} + (|J'| - 1)x_{i''} \leq |J'|\}. \tag{4.15e}$$

We aim to show that $\mathscr{P}_{\mathcal{IUC}}(G) = \mathscr{P}$. Validity of the set of inequalities (4.15) for the IUC polytope associated with G indicates that $\mathscr{P}_{\mathcal{IUC}}(G) \subseteq \mathscr{P}$. In order to prove that the reverse also holds, we show that every extreme point of $\mathscr{P}$ is integral, hence it is the incidence vector of an IUC in G, which implies $\mathscr{P} \subseteq \mathscr{P}_{\mathcal{IUC}}(G)$. The proof is by contradiction. Let $\hat{x}$ be a fractional extreme point of $\mathscr{P}$, and define $V_0 = \{v \in V \mid \hat{x}_v = 0\}$ and $V_1 = \{v \in V \mid \hat{x}_v = 1\}$. First, observe that every inequality in (4.15) involving x_v, for some $v \in V_0$, is redundant in description of the face $\mathscr{P}_{(x_{V_0}=0)} = \{x \in \mathscr{P} \,|\, x_v = 0, \forall v \in V_0\}$. To see this, consider a fixed vertex $\tilde{j} \in J \cap V_0$. If $|J| = 2$, i.e., $J = \{\tilde{j}, j'\}$, then (4.15) reduces to

$$\begin{aligned}\mathscr{P} = \{x \in \mathbb{R}^n \mid 0 \le x_{j'} \le 1;\ \ 0 \le x_i \le 1, \forall i \in I;\\ \sum_{i \in I'} x_i + (|I'| - 1)x_{j'} \le |I'|, \forall I' \subseteq I, |I'| \ge 2\},\end{aligned}$$

which is precisely the description of the IUC polytope associated with the star subgraph $G[\{j'\} \cup I]$. If $|J| \ge 3$, then each inequality of type (4.15b)-(4.15c) corresponding to $\{\tilde{j}, j'\}$ is dominated by an inequality of the same type corresponding to $\{j', j''\}, \forall j'' \in J \backslash \{\tilde{j}, j'\}$. Besides, each inequality of type (4.15d)-(4.15e) involving $x_{\tilde{j}}$ can be written as a linear combination of $x_{i'} \le 1$ (or $x_{i''} \le 1$) and another inequality of the same type corresponding to $J' \backslash \{\tilde{j}\}$. Therefore,

$$\begin{aligned}
\mathscr{P}_{(x_{V_0}=0)} = \{x \in \mathbb{R}^n \mid x_v = 0,\ \forall v \in V_0; \quad & (4.16a)\\
0 \le x_v \le 1,\ \forall v \in V \backslash V_0; \quad & (4.16b)\\
\forall \{j', j''\} \subseteq J \backslash V_0,\ \forall I' \subseteq I \backslash V_0, |I'| \ge 2: \quad & \\
\sum_{i \in I'} x_i + (|I'| - 1)x_{j'} + x_{j''} \le |I'|, \quad & (4.16c)\\
\sum_{i \in I'} x_i + x_{j'} + (|I'| - 1)x_{j''} \le |I'|; \quad & (4.16d)\\
\forall \{i', i''\} \subseteq I \backslash V_0,\ \forall J' \subseteq J \backslash V_0, |J'| \ge 2: \quad & \\
\sum_{j \in J'} x_j + (|J'| - 1)x_{i'} + x_{i''} \le |J'|, \quad & (4.16e)\\
\sum_{j \in J'} x_j + x_{i'} + (|J'| - 1)x_{i''} \le |J'|\}. \quad & (4.16f)
\end{aligned}$$

Now, consider the following three possibilities for the cardinality of V_1: $|V_1| = 0$, $|V_1| = 1$, and $|V_1| \geq 2$.

- CASE 1: $|V_1| = 0$ ($\hat{x}_v$ is fractional for every $v \in V \backslash V_0$.)

  Then, there exists a sufficiently small $\epsilon > 0$ for which $\hat{x}^+$ and $\hat{x}^-$ with

$$\hat{x}_i^+ = \hat{x}_i + \epsilon, \forall i \in I \backslash V_0;\ \hat{x}_j^+ = \hat{x}_j - \epsilon, \forall j \in J \backslash V_0;\ \hat{x}_v^+ = \hat{x}_v, \forall v \in V_0$$
$$\hat{x}_i^- = \hat{x}_i - \epsilon, \forall i \in I \backslash V_0;\ \hat{x}_j^- = \hat{x}_j + \epsilon, \forall j \in J \backslash V_0;\ \hat{x}_v^- = \hat{x}_v, \forall v \in V_0$$

  belong to $\mathscr{P}_{(x_{V_0}=0)} \subseteq \mathscr{P}$. Since $\hat{x} = \frac{1}{2}(\hat{x}^+ + \hat{x}^-)$, this contradicts extremity of $\hat{x}$.

- CASE 2: $|V_1| = 1$

  Without loss of generality, assume $V_1 = \{\tilde{j}\}$, for some $\tilde{j} \in J$. Note that, in description of the face $\mathscr{P}_{(x_{V_0}=0, x_{\tilde{j}}=1)}$, every inequality in (4.16d) corresponding to $\{\tilde{j}, j'\}, j' \in J \backslash (V_0 \cup \{\tilde{j}\})$, is dominated by the corresponding inequality of type (4.16c), as $x_{j'} \leq 1$. Furthermore, each inequality of type (4.16c) involving $x_{\tilde{j}}$ and corresponding to a proper subset of I is in turn dominated by the one corresponding to I itself, i.e.,

$$\sum_{i \in I} x_i + x_{j'} \leq 1,\ \forall j' \in J \backslash (V_0 \cup \{\tilde{j}\}). \tag{4.17}$$

  Now, it is clear that every inequality of type (4.16c)-(4.16d) corresponding to $\{j', j''\} \subseteq J \backslash (V_0 \cup \{\tilde{j}\})$ and I can be written as a linear combination of an inequality in (4.17), $x_{j'} \leq 1$, and $x_{j''} \leq 1$, and the ones corresponding to proper subsets of I are dominated by such linear combinations. Hence, (4.16c)-(4.16d) reduces to (4.17) in description of $\mathscr{P}_{(x_{V_0}=0, x_{\tilde{j}}=1)} = \{x \in \mathscr{P} \mid x_{\tilde{j}} = 1;\ x_v = 0, \forall v \in V_0\}$. Besides, in the description of this face, every inequality of type (4.16e)-(4.16f) corresponding to a set $J' = \tilde{J},\ \tilde{j} \notin \tilde{J}$, is dominated by the inequality corresponding to $J' = \tilde{J} \cup \{\tilde{j}\}$, thus (4.16e)-(4.16f) can be replaced by

$$\forall\{i',i''\} \subseteq I\backslash V_0, \forall J' \subseteq J\backslash(V_0 \cup \{\tilde{j}\}), |J'| \geq 1 :$$

$$\sum_{j\in J'} x_j + |J'|x_{i'} + x_{i''} \leq |J'|, \tag{4.18a}$$

$$\sum_{j\in J'} x_j + x_{i'} + |J'|x_{i''} \leq |J'|. \tag{4.18b}$$

Furthermore, all inequalities in (4.18) are dominated by linear combinations of the inequalities corresponding to $|J'| = 1$, i.e.,

$$x_{j'} + x_{i'} + x_{i''} \leq 1,\ \forall j' \in J\backslash(V_0 \cup \{\tilde{j}\}),\ \forall\{i',i''\} \subseteq I\backslash V_0, \tag{4.19}$$

and (4.19) is, in turn, dominated by (4.17). This implies that

$$\begin{aligned}\mathscr{P}_{(x_{V_0}=0,x_{\tilde{j}}=1)} = \{x \in \mathbb{R}^n \mid\ & x_{\tilde{j}} = 1;\ x_v = 0,\ \forall v \in V_0;\\ & 0 \leq x_v \leq 1,\ \forall v \in V\backslash(V_0 \cup \{\tilde{j}\});\\ & \sum_{i\in I} x_i + x_{j'} \leq 1,\ \forall j' \in J\backslash(V_0 \cup \{\tilde{j}\})\}.\end{aligned}$$

Let i' and i'' be two fixed vertices in I, and recall that $0 < \hat{x}_{i'}, \hat{x}_{i''} < 1$. Correspondingly, let $\hat{x}^+$ and $\hat{x}^-$ be the two distinct points defined as follows for a small-enough value of $\epsilon > 0$:

$$\begin{aligned}\hat{x}^+_{i'} = \hat{x}_{i'} + \epsilon;\ \hat{x}^+_{i''} = \hat{x}_{i''} - \epsilon;\ \hat{x}^+_v = \hat{x}_v, \forall v \in V\backslash\{i',i''\},\\ \hat{x}^-_{i'} = \hat{x}_{i'} - \epsilon;\ \hat{x}^-_{i''} = \hat{x}_{i''} + \epsilon;\ \hat{x}^-_v = \hat{x}_v, \forall v \in V\backslash\{i',i''\}.\end{aligned}$$

Then, $\hat{x} = \frac{1}{2}(\hat{x}^+ + \hat{x}^-)$ while $\hat{x}^+, \hat{x}^- \in \mathscr{P}_{(x_{V_0}=0,x_{\tilde{j}}=1)}$, which contradicts extremity of $\hat{x}$.

- CASE 3: $|V_1| \geq 2$

The set of inequalities (4.16) indicate that if $I \cap V_1 \neq \emptyset$ and $J \cap V_1 \neq \emptyset$, then each one of I and J must have exactly one vertex in V_1. Suppose $V_1 = \{i', j'\}$, for some $i' \in I$ and

$j' \in J$. Then, (4.16) immediately implies $\hat{x}_v = 0, \forall v \in V \backslash (V_0 \cup V_1)$, which contradicts $\hat{x}$ being fractional. On the other hand, if $I \cap V_1 = \emptyset$, then (4.16c)-(4.16d) imply $\hat{x}_i = 0, \forall i \in I$, and (4.16e)-(4.16f) become trivial and redundant. As a result,

$$\begin{aligned} \mathscr{P}_{(x_{V_0}=0, x_{V_1}=1)} = \{x \in \mathbb{R}^n \mid\ & x_v = 1, \forall v \in V_1; \\ & x_v = 0,\ \forall v \in V_0; \\ & 0 \le x_v \le 1,\ \forall v \in V \backslash (V_0 \cup V_1)\}, \end{aligned}$$

all extreme points of which are integral. Patently, the same result holds when $J \cap V_1 = \emptyset$, which contradicts fractionality of $\hat{x}$.

Hence, such a fractional extreme point $\hat{x}$ does not exist, and the proof is complete. □

It is clear that IUC, 1-dependent set, and co-1-defective clique are equivalent in complete bipartite graphs. This leads to the following corollary concerning the 1-dependent set and co-1-defective clique polytopes.

Corollary 20. *Let $G = K_{|I|,|J|}$ be a complete bipartite graph, where $|I|, |J| \ge 2$. Then, the family of inequalities* (4.10) *for all double-star subgraphs of G, together with the variable bounds, is sufficient to describe the* 1*-dependent set (equivalently, co-1-defective clique) polytope associated with G.*

4.3.3 Fan and wheel

A *fan* graph on n vertices is composed of one vertex adjacent to $n-1$ vertices inducing a path (chain) subgraph. Following our notation in the previous section, we denote the vertex set of a fan graph by $F = \{h\} \cup P$, where h is the hub vertex and P denotes the vertex set of the corresponding path subgraph. Recall that a path graph itself is not facet-producing for the IUC polytope by Theorem 17. Conventionally, we assume that the edges of $G[P]$ are given by $\{i, i+1\}, \forall i \in \{1, \ldots, |P|-1\}$.

Theorem 21. (Fan Inequality) *Let $F = \{h\} \cup P$, $|P| \geq 4$, be a subset of vertices inducing a fan graph in $G = (V, E)$. Given $|P| = 3q + r$, where q is a positive integer and $r \in \{0, 1, 2\}$, the inequality*

$$\sum_{i \in P} x_i + \left(2(q-1) + \left\lfloor \frac{2(r+1)}{3} \right\rfloor \right) x_h \leq 2q + \left\lfloor \frac{2(r+1)}{3} \right\rfloor \tag{4.20}$$

is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$, and induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[F])$ if and only if $r \neq 2$.

Proof. First, note that the right-hand side of (4.20) is the cardinality of a maximum IUC in $G[P]$; hence, the inequality

$$\sum_{i \in P} x_i \leq 2q + \left\lfloor \frac{2(r+1)}{3} \right\rfloor \tag{4.21}$$

is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$. More formally, (4.21) is the Gomory-Chvátal cut corresponding to the inequality obtained from the summation of all OT inequalities of $G[P]$ and $2x_1 + x_2 + x_{|P|-1} + 2x_{|P|} \leq 6$ divided by 3. Besides, every maximal IUC containing h in $G[F]$ is a clique of size 3, since every pair of non-adjacent vertices in $G[P]$ with h induces an open triangle in this subgraph. Therefore, $\max\{\sum_{i \in P} x_i \mid x \in \mathscr{P}_{\mathcal{IUC}}(G[F]) \text{ and } x_h = 1\} = 2$, and the coefficient of x_h due to lifting into (4.21) is $2q + \lfloor \frac{2(r+1)}{3} \rfloor - 2$. This establishes validity of (4.20).

Next, we show that $r \neq 2$ is sufficient for (4.20) to induce a facet of the IUC polytope associated with $G[F]$. For $r = 0$, consider the proper face $\mathcal{F} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G[F]) \mid \sum_{i \in P} x_i + 2(q-1)x_h = 2q\}$ of $\mathscr{P}_{\mathcal{IUC}}(G[F])$ and an arbitrary supporting hyperplane $\mathscr{H} : a_h x_h + \sum_{i \in P} a_i x_i = b$ that contains $\mathcal{F}$. Let x_{U_1} and x_{U_2} be the incidence vectors of two maximum IUCs in $G[P]$ given as follows:

$$\begin{aligned} U_1 &= \{1, 3, 3p-1, 3p, \forall p \in \{2, \ldots, q\}\}, \\ U_2 &= \{2, 3, 3p-1, 3p, \forall p \in \{2, \ldots, q\}\}. \end{aligned} \tag{4.22}$$

These are the sets of black vertices in the path graphs obtained from eliminating the vertex in the last partition in the graphs of Figure 4.1. Clearly, $x_{U_1}, x_{U_2} \in \mathcal{F} \subseteq \mathscr{H}$, which immediately leads to $a_1 = a_2$. In addition, $a_i = a_{i+2}, \forall i \in \{1, \ldots, |P|-2\}$, as the incidence vector of every clique $\{h, i, i+1\}, \forall i \in \{1, \ldots, |P|-1\}$, in $G[F]$ belongs to $\mathcal{F} \subseteq \mathscr{H}$. This indicates that $a_i = a, \forall i \in P$, $b = 2aq$, and $a_h = 2a(q-1)$. Finally, $\mathscr{P}_{\mathcal{IUC}}(G[F])$ having non-empty interior implies $a \neq 0$, so $\mathcal{F}$

is a facet of $\mathscr{P}_{\mathcal{IUC}}(G[F])$. For the case $r = 1$, using the same argument based on

$$U_1 = \{3q+1, 1, 2, 3p-2, 3p, \forall p \in \{2, \ldots, q\}\},$$
$$U_2 = \{3q+1, 1, 3, 3p-2, 3p, \forall p \in \{2, \ldots, q\}\},$$

we may show that the inequality $\sum_{i \in P} x_i + (2q-1)x_h \leq 2q+1$ is facet-inducing for $\mathscr{P}_{\mathcal{IUC}}(G[F])$. This completes the sufficiency part of the proof.

It remains to prove that (4.20) is not facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G[F])$ if $r = 2$. Observe that $G[F]$ has exactly $|P| - 1$ IUCs containing h whose incidence vectors belong to $\mathcal{F} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G[F]) \mid \sum_{i \in P} x_i + 2qx_h = 2q + 2\}$, that is $\{h, i, i+1\}, \forall i \in \{1, \ldots, |P| - 1\}$. Thus, every other extreme point of $\mathscr{P}_{\mathcal{IUC}}(G[F])$ that lies on $\mathcal{F}$ must be the incidence vector of a maximum IUC in $G[P]$. On the other hand, if $r = 2$, $G[P]$ has a unique maximum IUC, whose incidence vector satisfies $x_1 = x_2 = x_{|P|-1} = x_{|P|} = 1$. Therefore, exactly $|P|$ extreme points of $\mathscr{P}_{\mathcal{IUC}}(G[F])$ belong to $\mathcal{F}$, which implies *dim*$(\mathcal{F}) \leq |P| - 1 = |F| - 2$. Recall that, every point $x \in \mathcal{F}$ is a convex combination of these extreme points. Hence, $\mathcal{F}$ is not a facet of $\mathscr{P}_{\mathcal{IUC}}(G[F])$, and the proof is complete. □

Every fan graph contains quadratically many induced subgraphs of the same type, most of which satisfy the conditions of Theorem 21. The following theorem states that the fan inequalities corresponding to those subgraphs are facet-inducing for the IUC polytope associated with the supergraph.

Theorem 22. *Let $G = (F, E)$, $F = \{h\} \cup P$, be a fan graph. Then, every fan inequality* (4.20) *corresponding to $F' = \{h\} \cup P'$, $P' \subseteq P$, satisfying the cardinality condition of Theorem 21 induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G)$.*

Proof. Without loss of generality, let $P' = \{j, j+1, \ldots, k-1, k\}$ and consider lifting a variable $x_i, i \in P \backslash P'$ into the fan inequality (4.20) corresponding to $F' = \{h\} \cup P'$. Note that $P' \cap N(i) = \emptyset, \forall i \in P \backslash \{j-1, \ldots, k+1\}$, thus every maximum IUC in $G[P']$ together with $i \in P \backslash \{j - 1, \ldots, k+1\}$ forms an IUC in G. Besides, observe in the proof of Theorem 21 that $G[P']$ always has a maximum IUC in which j is an isolated vertex. As $P' \cap N(j-1) = \{j\}$, the union of this set and the vertex $j-1$ is also an IUC in G. By symmetry, the same result holds for the vertex $k+1$. This implies that the coefficient of every variable $x_i, \forall i \in P \backslash P'$, vanishes in an arbitrary sequential lifting of the fan inequality corresponding to F', which completes the proof. □

Next, we present similar results concerning the IUC polytope associated with a *wheel* graph. A wheel graph induced by $W = \{h\} \cup H$ consists of a hub vertex h connected to all vertices of a hole H. As before, we assume that adjacent vertices in $G[H]$ have consecutive labels, and $|H| + 1 \equiv 1$. Novelty of the following result mainly concerns the case that $|H|$ is a multiple of 3.

Theorem 23. (Wheel Inequality) *Let $W = \{h\} \cup H,\ |H| \geq 4$, be a subset of vertices inducing a wheel graph in $G = (V, E)$. Given $|H| = 3q + r$, where q is a positive integer and $r \in \{0, 1, 2\}$, the inequality*

$$\sum_{i \in H} x_i + \left(2(q-1) + \left\lfloor \frac{2r}{3} \right\rfloor\right) x_h \leq 2q + \left\lfloor \frac{2r}{3} \right\rfloor \tag{4.23}$$

is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$, and induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[W])$ if and only if $r \neq 0$ or q is odd.

Proof. Validity of (4.23) as well as its facial property when $r \neq 0$ are evident, as (4.23) is obtained from lifting x_h into the hole inequality (4.7). Similar to a fan graph, $\max\{\sum_{i \in H} x_i \mid x \in \mathscr{P}_{\mathcal{IUC}}(G[W]) \text{ and } x_h = 1\} = 2$, hence the coefficient of x_h in such a lifted hole inequality is $2q + \lfloor \frac{2r}{3} \rfloor - 2$.

Let $r = 0$, and consider $\mathcal{F} = \{x \in \mathscr{P}_{\mathcal{IUC}}(G[W]) \mid \sum_{i \in H} x_i + 2(q-1)x_h = 2q\}$ as well as a supporting hyperplane $\mathscr{H} : a_h x_h + \sum_{i \in H} a_i x_i = b$ of $\mathscr{P}_{\mathcal{IUC}}(G[W])$ containing it. The incidence vector of every clique $\{h, i, i+1\}, \forall i \in \{1, \ldots, |H|\}$, in $G[W]$ belongs to $\mathcal{F} \subseteq \mathscr{H}$, thus $a_i = a_{i+2}, \forall i \in \{1, \ldots, |H|\}$. If q is odd—equivalently, H is of odd cardinality—this result further implies that $a_{|H|} = a_2$, which immediately leads to $a_i = a, \forall i \in \{1, \ldots, |H|\}$, $b = 2aq$, and $a_h = 2a(q-1)$. Since the interior of $\mathscr{P}_{\mathcal{IUC}}(G[W])$ is non-empty, $a \neq 0$ and the sufficiency part of the proof is complete.

To prove the necessity, we examine the case where $r = 0$ and q is even. Similar to the proof of Theorem 21, our argument is based on the number of extreme points of $\mathscr{P}_{\mathcal{IUC}}(G[W])$ that lie on $\mathcal{F}$, and the dimension of the affine subspace containing them. Patently, $G[W]$ has exactly $|H|$ IUCs containing h whose incidence vectors belong to $\mathcal{F}$, i.e., $\{h, i, i+1\}, \forall i \in \{1, \ldots, |H|\}$. Recall from Theorem 9 that the right-hand-side of (4.23) is the IUC number of $G[H]$, hence every other extreme point of $\mathscr{P}_{\mathcal{IUC}}(G[W])$ lying on $\mathcal{F}$ must be the incidence vector of a maximum IUC in $G[H]$. It is easy to verify that, if $|H|$ is a multiple of 3, $G[H]$ always has three distinct maximum IUCs (of cardinality $2q$) as follows:

$$U_1 = \{3p-2, 3p-1, \forall p \in \{1, \ldots, q\}\},$$
$$U_2 = \{3p-1, 3p, \forall p \in \{1, \ldots, q\}\},$$
$$U_3 = \{3p-2, 3p, \forall p \in \{1, \ldots, q\}\}.$$

Hence, exactly $|H|+3$ extreme points of $\mathscr{P}_{\mathcal{IUC}}(G[W])$ belong to $\mathcal{F}$.

It is known that the adjacency matrix of a cycle graph with even number of vertices is rank deficient, which immediately implies that the affine subspace of $\mathbb{R}^{|H|}$ spanned by the points $x_{\{i,i+1\}} = e_i + e_{i+1}, \forall i \in \{1, \ldots, |H|\}$, is at most $|H|-2$ dimensional. Besides, the incidence vectors of $U_1, U_2,$ and U_3 in $\mathbb{R}^{|H|}$ can be easily written as linear combinations of these points, hence the dimension of the subspace of $\mathbb{R}^{|H|}$ spanned by $\{x_{U_1}, x_{U_2}, x_{U_3}, x_{\{i,i+1\}}, \forall i \in \{1, \ldots, |H|\}\}$, is no more than $|H|-2$. Since all extreme points of $\mathscr{P}_{\mathcal{IUC}}(G[W])$ lying on $\mathcal{F}$ are generated from a point in this set by adding the coordinate corresponding to h, the subspace of $\mathbb{R}^{|W|}$ spanned by them is at most $|H|-1 = |W|-2$ dimensional. Finally, because every point of $\mathcal{F}$ is a convex combination of these extreme points, dimension of this face is no more than $|W|-2$, therefore $\mathcal{F}$ is not a facet. The proof is complete. $\square$

The result of Theorem 22 naturally extends to the induced fans of a wheel graph. The only exception is given by the subgraphs obtained from eliminating a single vertex (and incident edges) from the corresponding chordless cycle. Let $F = \{h\} \cup P$, where $P = H \backslash \{v\}$, be the vertex set of such a fan subgraph, while $|P|$ satisfies the facet-defining conditions of Theorem 21, that is $|P| = 3q+1$ or $|P| = 3q$. Observe that, in this case, $|H|$ also satisfies the facet-defining conditions of Theorem 23, that is $|H| = 3q + 2$ or $|H| = 3q + 1$. Then, it can be easily shown that the wheel inequality (4.23) corresponding to $W = \{h\} \cup H$ coincides with the lifted fan equality (4.20) corresponding to $F = \{h\} \cup P$.

Patently, this result also applies to a *defective wheel* graph. A defective wheel is obtained from a (complete) wheel graph by eliminating some edges connecting the hub vertex to the corresponding hole. In addition to its fan subgraphs, a defective wheel always has at least one chordless cycle that contains its hub vertex. Examples of defective wheels are given in Figure 4.3. Let H_h denote a hole containing the hub vertex h in a defective wheel graph. In reference to Theorem 12 of Section 4.3.1, observe that every vertex not in H_h has at most two neighbors in this set. The following corollary summarizes the implications of Theorems 22 and 12 regarding the facial structure of the IUC polytope associated with a defective wheel graph.

Corollary 24. *Let $G = (V, E)$ be a defective wheel graph. Then,*

(i) *inequality* (4.20) *corresponding to a fan subgraph satisfying the cardinality condition of Theorem 21 is facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$. Besides,*

(ii) *inequality* (4.7) *corresponding to a hole containing the hub vertex that satisfies the cardinality condition of Theorem 9 induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G)$.*

Finally, note that the IUC polytope associated with a fan graph, likewise a wheel graph, possesses exponentially many facets induced by the star inequality (4.9) as well.

Lifting fan and wheel inequalities

We consider lifting the fan and wheel inequalities under the conditions that they are facet-defining for the IUC polytope associated with the corresponding induced subgraph. Similar to Section 4.3.1, our focus is on the distribution of $N(v), v \in V \backslash F$ (resp. $v \in V \backslash W$), within $G[F]$ (resp. $G[W]$) and the cases where the lifting coefficients can be determined without solving a computationally expensive lifting problem.

First, observe that the result of Theorem 12, concerning the hole inequality (4.7), also applies to the wheel inequality (4.23) since an IUC of cardinality $2q + \lfloor \frac{2r}{3} \rfloor + 1$ in $G[W \cup \{v\}], v \in V \backslash W$, can always be attained through the vertices of H (together with v) under the theorem conditions. However, the result concerning redundancy of the distribution of neighbors of v does not extend to the fan inequality (4.20) as well as the wheel inequality (4.23) when $r = 0$ (and q is odd). Consider, for example, the case where $|P| = 7$ and $P \cap N(v) = \{1, 3\}$, or $|H| = 9$ and $H \cap N(v) = \{1, 5\}$. In addition, it is trivial that $|P \cap N(v)| \leq 1$ (resp. $|H \cap N(v)| \leq 1$) is sufficient for the coefficient of $x_v, v \in V \backslash F$ (resp. $v \in V \backslash W$), to vanish in the latter cases, regardless of the role of the hub vertex h. In general, however, the lifting coefficient of a variable x_v is highly influenced by connectivity of v and h. Particularly, if $\{h, v\} \in E$, existence of two adjacent vertices in $P \cap N(v)$ (resp. $H \cap N(v)$) is sufficient for the corresponding lifting coefficient to vanish. On the other hand, if $\{h, v\} \notin E$, existence of two adjacent vertices in $P \backslash N(v)$ (resp. $H \backslash N(v)$) brings about the same result. The following theorem concerns redundancy of the distribution of $N(v)$ within $G[F]$ given the contribution of the hub vertex h. Note that, the coefficient of x_v in every sequential lifting of the corresponding inequality will not exceed 2 because $\{h, v\}$ is always an IUC in the graph.

Theorem 25. *Consider the fan inequality* (4.20) *where $r \neq 2$. In every sequential lifting of this inequality, the coefficient of a variable x_v, $v \in V \backslash F$, vanishes under the following conditions:* (i) *$h \in N(v)$ and $|P \cap N(v)| \geq \left\lceil \frac{|P|}{2} \right\rceil + 1$, or* (ii) *$h \notin N(v)$ and $|P \cap N(v)| \leq \left\lfloor \frac{|P|}{2} \right\rfloor - 1$.*

Proof. It suffices to observe that the cardinality of $P \cap N(v)$ in each case guarantees the existence of two adjacent vertices $i, i+1 \in P$ such that $\{h, v, i, i+1\}$ is an IUC in $G[F \cup \{v\}]$. In the former case, this IUC is a clique, and in the latter, it is the union of a clique and an isolated vertex. $\square$

A similar result holds for the wheel inequality, which we skip here. We finish this section with presenting a family of facet-defining inequalities for the IUC polytope associated with an anti-cycle graph due to its fan substructures. We assume the vertices of an anti-cycle graph are labeled as described in the proof of Theorem 11, and $|A| + 1 \equiv 1$.

Theorem 26. *Let $G = (A, E)$ be an anti-cycle graph on $|A| \geq 8$ vertices. Then, the inequality*

$$\sum_{j=i}^{i+3} x_j + x_k + x_{k+1} \leq 3, \ \forall i \in A, \ \forall k \in \{i + 4, \ldots, |A| + i - 2\}, \tag{4.24}$$

induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G)$.

Proof. Observe that $P = \{i, i+1, i+2, i+3\}, \forall i \in A$, induces a path (chain) graph on 4 vertices, and $G[\{k\} \cup P], \forall k \in \{i+5, \ldots, |A|+i-2\}$, is a fan in G. In fact, for a fixed vertex i, the vertices $i+4$ and $|A|+i-1$ are the only ones in $A \backslash P$ which are not adjacent to all vertices of P. Therefore, the inequality $\sum_{j=i}^{i+3} x_j + x_k \leq 3, \forall k \in \{i+5, \ldots, |A|+i-2\}$, is valid for $\mathscr{P}_{\mathcal{IUC}}(G)$ and induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G[\{k\} \cup P])$, by Theorem 21. First, consider lifting the variable x_{k+1} into $\sum_{j=i}^{i+3} x_j + x_k \leq 3$, for some $k \in \{i+5, \ldots, |A|+i-3\}$. Since $\{k, k+1\} \notin E$ and $P \cap N(k+1) = P$, the coefficient of x_{k+1} in the corresponding lifted inequality will be 1 and the inequality $\sum_{j=i}^{i+3} x_j + x_k + x_{k+1} \leq 3$ is facet defining for $\mathscr{P}_{\mathcal{IUC}}(G[\{k, k+1\} \cup P])$. Besides, every other vertex $v \in A \backslash (\{k, k+1\} \cup P)$ has at least one neighbor in $\{k, k+1\}$ and $|P \cap N(v)| \geq 3$, hence the coefficient of x_v vanishes in every sequential lifting of this inequality, by Theorem 25. Therefore, the inequality $\sum_{j=i}^{i+3} x_j + x_k + x_{k+1} \leq 3$ induces a facet of $\mathscr{P}_{\mathcal{IUC}}(G)$, for every $k \in \{i+5, \ldots, |A|+i-3\}$. It remains to show that, for every $i \in A$, the inequalities $\sum_{j=i}^{i+3} x_j + x_{i+4} + x_{i+5} \leq 3$ and $\sum_{j=i}^{i+3} x_j + x_{|A|+i-2} + x_{|A|+i-1} \leq 3$ are facet-defining for $\mathscr{P}_{\mathcal{IUC}}(G)$. We present the proof for the former inequality, i.e.,

$$\sum_{j=i}^{i+5} x_j \leq 3, \tag{4.25}$$

for an arbitrary vertex i; the latter then follows by symmetry of the graph structure. It is easy to verify that (4.25) is obtained from lifting x_{i+4} into the fan inequality $\sum_{j=i}^{i+3} x_j + x_{i+5} \leq 3$, hence facet-defining for the IUC polytope associated with the corresponding induced subgraph. Besides, every vertex $k \in \{i+7, \ldots, |A|+i-1\}$ is adjacent to the vertex $i+5$, thus the corresponding coefficient vanishes in lifting x_k into (4.25), by Theorem 25. Finally, observe that $\{i, i+2, i+4, i+6\}$ is a clique in G, therefore, $\max\{\sum_{j=i}^{i+5} x_j \mid x \in \mathscr{P}_{\mathcal{IUC}}(G) \text{ and } x_{i+6} = 1\} = 3$, and the coefficient of x_{i+6} also vanishes in lifting this variable into (4.25). The proof is complete. $\square$

Note that the anti-cycle graph on $|A| = 6$ vertices does not have an induced (complete) fan, and the inequality $\sum_{j=1}^{6} x_j \leq 3$ is not facet-inducing for the corresponding IUC polytope, previously shown by Theorem 11. Also, for the anti-cycle graph on $|A| = 7$ vertices, the inequalities $\sum_{j=1}^{6} x_j \leq 3$ and $\sum_{j=1}^{4} x_j + x_6 + x_7 \leq 3$ are both dominated by the anti-hole inequality $\sum_{j=1}^{7} x_j \leq 3$ of Theorem 11.

4.4 The Separation Problems

In this section, we study the computational complexity of the separation problem for each family of the valid inequalities presented in the previous section for the IUC polytope. Given a family of (linear) inequalities $\mathscr{F}$ and a point x^*, the corresponding separation problem is to decide whether x^* belongs to the polyhedron defined by $\mathscr{F}$, i.e., if x^* satisfies all inequalities in $\mathscr{F}$, and if not, find an inequality which is violated by x^*. Efficient separating subroutines are essential in devising branch-and-cut algorithms for integer (linear) programs; however, our results in this part are negative in that we can show that the separation problems for most of the proposed valid inequalities for the IUC polytope are NP-hard.

Complexity of the separation problem for the family of hole, as well as anti-hole, inequalities for the IUC polytope is yet to be determined, although there is a rich body of literature concerning identification of holes and anti-holes in graphs. In fact, the focus of this line of research has been mostly on holes and anti-holes of odd cardinality due to the Berge's strong perfect graph conjecture [115]. According to this conjecture, a graph G is perfect, i.e., the chromatic number of every induced subgraph of G equals its clique number, if and only if G does not contain an odd hole or odd anti-hole. Chudnovsky et al. [116] proved this conjecture, and Chudnovsky et al. [117] proposed an $\mathcal{O}(n^9)$-time algorithm to test if a graph has an odd hole or odd anti-hole; hence deciding if it is perfect. However, it remained an open problem whether or not it is polynomial-time decidable if a graph has an odd hole (equivalently, an odd anti-hole) until very recently that Chudnovsky et al. [118] presented an $\mathcal{O}(n^9)$-time algorithm for it. It is also known that the problem of deciding if a graph has a hole (equivalently, an anti-hole) of even cardinality is tractable [119], and the most efficient algorithm proposed for this problem has a time complexity of $\mathcal{O}(n^{11})$ [120]. On the other hand, it is NP-complete to decide if a graph has an odd (resp. even) hole containing a given vertex [121, 122, 120]. It is clear—due to feasibility of $x^* = \frac{2}{3}\mathbf{1}$ to the fractional IUC polytope—that the separation problem for the family of hole inequalities (4.7) for the IUC polytope is at least as hard as determining if a graph has a hole whose cardinality is not a multiple of 3, for which we are not aware of an efficient algorithm. Similarly, the feasibility of $x^* = \frac{1}{2}\mathbf{1}$ to the fractional IUC polytope implies that the separation problem for the family of anti-hole inequalities (4.8) for the IUC polytope is at least as hard as deciding if a graph has an odd hole. It should be mentioned that the separation problem for the family of odd-hole inequalities for the vertex packing polytope (equivalently, odd-anti-hole inequalities for the clique polytope) is polynomial-time solvable [75]; however, this result does not automatically extend to the family of odd-anti-hole inequalities for the IUC polytope.

In the rest of this section, we show that the separation problems for the families of star, double-star, fan and wheel inequalities for the IUC polytope are all NP-hard.

Theorem 27. *The separation problem for the family of star inequalities* (4.9) *for the IUC polytope is NP-hard.*

Proof. The reduction is from the INDEPENDENT SET problem (k-IS), which asks if a simple, undirected graph $G = (V, E)$ has an independent set of cardinality at least k. Without loss of generality, we assume $k \geq 3$. Corresponding to an instance (G, k) of k-IS, consider an instance (G', x^*) of the separation problem for the family of star inequalities (4.9), where $G' = (V', E')$ is constructed by appending a new vertex h to G adjacent to all vertices of V, i.e.,

$$V' = V \cup \{h\},$$
$$E' = E \cup \{\{h, i\}, \forall i \in V\},$$

and x^* is defined as follows:

$$x_h^* = 1,$$
$$x_i^* = \frac{1}{k-1}, \ \forall i \in V.$$

Note that $k \geq 3$ implies $x^* \in \mathscr{P}_{\mathcal{IUC}}^F(G')$. We claim that (G, k) is a positive instance of k-IS if and only if x^* violates a star inequality in G'. Let I^* be an independent set of cardinality k in G. Then,

$S = \{h\} \cup I^*$ induces a star subgraph in G', and

$$\sum_{i \in I^*} x_i^* + (|I^*| - 1)x_h^* = \frac{k}{k-1} + (k-1) = k + \frac{1}{k-1} > k = |I^*|,$$

which indicates violation of the star inequality associated with $G'[S]$. To prove the reverse, suppose that x^* violates the inequality associated with a star subgraph induced by $S' = \{v\} \cup I'$ in G', for some $I' \subseteq V'$ and $v \in V' \backslash I'$. Evidently, $I' \subseteq V$; since $x_v^* \leq 1 = x_h^*, \forall v \in V'$, this also indicates violation of the star inequality associated with $G'[\{h\} \cup I']$, i.e.,

$$\sum_{i \in I'} x_i^* + (|I'| - 1)x_h^* = \frac{|I'|}{k-1} + (|I'| - 1) > |I'|,$$

which implies $|I'| > k - 1$, or equivalently $|I'| \geq k$. The proof is complete. $\square$

Theorem 28. *The separation problem for the family of double-star inequalities* (4.10) *for the IUC polytope is NP-hard.*

Proof. The proof is similar to Theorem 27. Corresponding to an instance (G, k) of k-IS, consider an instance $(G' = (V', E'), x^*)$ of the separation problem for the family of double-star inequalities (4.10) constructed as follows:

$$\begin{aligned}
&V' = V \cup \{h, u\}, \\
&E' = E \cup \{\{h, i\}, \forall i \in V\} \cup \{\{u, i\}, \forall i \in V\}, \\
&x_h^* = 1, \\
&x_i^* = \frac{1}{k}, \ \forall i \in V \cup \{u\}.
\end{aligned}$$

Then, the same argument leads to the fact that (G, k) is a positive instance of k-IS if and only if x^* violates a double-star inequality in G'. $\square$

Theorem 29. *The separation problem for the family of fan inequalities* (4.20) *for the IUC polytope is NP-hard.*

Proof. We present a reduction from the INDUCED PATH problem (k-IP), which is to decide if a simple, undirected graph $G = (V, E)$ has an induced path subgraph (equivalently, a chordless path) on at least k vertices. Similar to k-IS, this problem is among the classical NP-complete problems listed by Garey and Johnson [123]. The proof idea is similar to that for Theorem 27. Let $k \geq 5$, and corresponding to an instance (G, k) of k-IP, consider an instance $(G' = (V', E'), x^*)$ of the separation problem for the family of fan inequalities (4.20) constructed as follows:

$$\begin{aligned}
&V' = V \cup \{h\},\\
&E' = E \cup \{\{h, i\}, \forall i \in V\},\\
&x_h^* = 1,\\
&x_i^* = \frac{2}{k-1}, \ \forall i \in V.
\end{aligned}$$

Let P^* be the vertex set of a chordless path in G on $k = 3q_k + r_k$ vertices. Then, $F = \{h\} \cup P^*$ induces a fan subgraph in G', and

$$\begin{aligned}
\sum_{i \in P^*} x_i^* + \left(2(q_k - 1) + \left\lfloor \frac{2(r_k+1)}{3} \right\rfloor\right) x_h^* &= \frac{2k}{k-1} + 2q_k - 2 + \left\lfloor \frac{2(r_k+1)}{3} \right\rfloor \\
&= \frac{2}{k-1} + 2q_k + \left\lfloor \frac{2(r_k+1)}{3} \right\rfloor \\
&> 2q_k + \left\lfloor \frac{2(r_k+1)}{3} \right\rfloor,
\end{aligned}$$

which indicates violation of the fan inequality associated with $G'[F]$. Now, suppose that x^* violates a fan inequality in G'. Given the magnitude of the components of x^*, this also implies violation of the inequality associated with a fan subgraph $G'[\{h\} \cup P']$, for some $P' \subseteq V$, i.e.,

$$\sum_{i \in P'} x_i^* + \left(2(q' - 1) + \left\lfloor \frac{2(r'+1)}{3} \right\rfloor\right) x_h^* > 2q' + \left\lfloor \frac{2(r'+1)}{3} \right\rfloor,$$

where $|P'| = 3q' + r'$. This leads to

$$\sum_{i \in P'} x_i^* = \frac{2|P'|}{k-1} > 2,$$

or equivalently, $|P'| \geq k$, and completes the proof. $\square$

Theorem 30. *The separation problem for the family of wheel inequalities* (4.23) *for the IUC polytope is NP-hard.*

Proof. The proof is almost identical to the proof of Theorem 29. The difference is that, instead of k-IP, the reduction is from the INDUCED CYCLE problem, which is also known to be NP-complete [123]. The INDUCED CYCLE problem (k-IC) is to decide if a simple, undirected graph $G = (V, E)$ has a chordless cycle on at least k vertices. Then, using the same argument on an instance $(G' = (V', E'), x^*)$ of k-IC constructed as stated in the proof of Theorem 29, one can show that the separation problem for the family of wheel inequalities (4.23) is NP-hard. $\square$

It is worth noting that, from the viewpoint of parameterized complexity, the INDEPENDENT SET, INDUCED PATH, and INDUCED CYCLE problems—parameterized by the solution size—are all W[1]-complete [124], implying that it is unlikely that there exists a fixed-parameter-tractable (fpt) algorithm for either of them. This result immediately implies that the separation problems for the families of star, double-star, fan and wheel inequalities remain intractable even if the separation problem is defined with respect to a subset of inequalities from the corresponding family with a fixed (small) number of variables.

4.5 Computational Experiments

In this section, we present the results of our computational experiments. We have conducted two sets of experiments to evaluate the effectiveness of incorporating the proposed valid inequalities in integer (linear) programming solution approaches for the maximum IUC problem.

In the first set of experiments, we investigated the effectiveness of each class of valid inequalities, as well as their combinations, using a set of graphs whose structures were known *a priori*. We generated a set of 20 undirected graphs, each of which is essentially a collection of 21 *principal* subgraphs that are sparsely connected to each other. Every principal subgraph of each instance has been randomly selected to be a cycle, anti-cycle, star, double-star, fan or wheel, and its number of vertices is given by the floor of a number drawn from a normal distribution with mean 10 and standard deviation 1. This choice of parameters yields graphs with around 200 vertices. The principal subgraphs have been connected to one another by a sparse set of random edges. To do this, an edge has been generated between every pair of vertices from two different principal subgraphs with a probability 0.01 independent of the others. Table 4.1 displays the characteristics of these instances, including the number of open triangles $|\Lambda|$ and the number of principal subgraphs for each structure type. In this table, "A-cycle" and "D-star" stand for anti-cycle and double-star, respectively.

Table 4.1: First set of experiments: characteristics of the test instances.

				Number of principal subgraphs					
Name	$\|V\|$	$\|E\|$	$\|\Lambda\|$	Cycle	A-cycle	Star	D-star	Fan	Wheel
`rnd#1`	204	550	2,464	3	2	2	3	7	4
`rnd#2`	213	630	2,857	2	4	6	1	3	5
`rnd#3`	196	510	2,273	3	2	3	5	6	2
`rnd#4`	194	470	2,056	6	2	3	3	2	5
`rnd#5`	211	623	3,078	4	5	5	3	3	1
`rnd#6`	194	617	2,673	2	8	2	4	2	3
`rnd#7`	193	488	2,014	2	3	3	4	6	3
`rnd#8`	198	506	2,226	5	2	4	4	4	2
`rnd#9`	194	496	2,293	3	2	5	3	3	5
`rnd#10`	194	472	1,998	2	3	5	6	3	2
`rnd#11`	203	567	2,644	4	3	1	5	5	3
`rnd#12`	188	466	2,143	2	1	2	5	8	3
`rnd#13`	202	514	2,537	4	2	4	6	4	1
`rnd#14`	212	497	2,070	7	2	5	2	3	2
`rnd#15`	206	575	2,536	4	3	3	2	5	4
`rnd#16`	201	550	2,608	2	3	2	7	2	5
`rnd#17`	204	586	2,751	4	4	3	5	2	3
`rnd#18`	200	581	2,813	3	5	4	3	2	4
`rnd#19`	204	562	2,422	5	4	2	1	5	4
`rnd#20`	203	550	2,268	5	3	2	3	7	1

We solved the maximum IUC problem on each instance using the base formulation (4.5), i.e., considering only the OT inequalities, as well as using this formulation strengthened by the valid inequalities corresponding to its principal subgraphs. For each principal subgraph, we generated only one inequality, except for the double-star subgraphs where both inequalities of (4.10) were generated. Recall that anti-cycle, star, double-star, fan and wheel graphs may generate several (potentially facet-defining) valid inequalities of the same or different types; yet, we restricted the incorporated inequalities to the whole structures, i.e., the principal subgraphs themselves, so we could have a better measure of the effectiveness of each family. We performed the experiments using ILOG/CPLEX 12.7 solver on a Dell Precision Workstation T7500® machine with eight 2.40 GHz Intel Xeon® processors and 12 GB RAM. We used the default settings of the solver except for the automatic cut generation, which was off in our first set of experiments.

Table 4.2 presents the solution time for each instance using the base formulation (4.5) as well as the strengthened formulations by the individual classes of the proposed valid inequalities. In this table, "opt." denotes the optimal solution value for each instance. The column "Base" reports the solution time under the base formulation (4.5). The next three columns, named "+HA", "+SD", and "+FW", present the solution time for each instance when the base formulation is strengthened by the hole (4.7) and anti-hole (4.8) inequalities, star (4.9) and double-star (4.10) inequalities, and fan (4.20) and wheel (4.23) inequalities associated with its principal subgraphs, respectively. The last three columns of this table show the percentage improvement in the solution time for each instance per appended valid inequality; that is,

$$\frac{\text{Base solution time} - \text{Improved solution time}}{\text{Base solution time} \times \text{Number of incorporated inequalities}} \times 100.$$

The boldface numbers in these columns show the highest improvement for each instance. Note that the highest per-inequality improvement does not necessarily imply the best solution time, due to different number of principal subgraphs of each type. On 10 instances (out of 20), the most per-inequality improvement was attained by the FW family (fan and wheel inequalities). With a

Table 4.2: First set of experiments: improvement (in solution time) for each family of the valid inequalities.

		Solution time (CPU sec.)				Impv. per ineq. (%)		
Name	opt.	Base	+HA	+SD	+FW	+HA	+SD	+FW
`rnd#1`	111	3,094.30	989.39	1,804.45	715.48	**13.6**	5.2	7.0
`rnd#2`	118	1,103.08	765.27	791.47	369.09	5.1	3.5	**8.3**
`rnd#3`	114	90.77	87.08	92.72	21.45	0.8	−0.2	**9.5**
`rnd#4`	110	197.38	180.75	177.79	108.90	1.1	1.1	**6.4**
`rnd#5`	118	1,218.80	471.90	510.82	486.17	6.8	5.3	**15.0**
`rnd#6`	103	9,820.76	250.19	5,012.75	2,299.58	9.7	4.9	**15.3**
`rnd#7`	110	173.19	61.19	120.13	82.52	**12.9**	2.8	5.8
`rnd#8`	115	115.95	54.36	75.58	87.51	**7.6**	2.9	4.1
`rnd#9`	113	95.45	83.50	51.37	22.74	2.5	4.2	**9.5**
`rnd#10`	114	48.18	11.76	43.24	18.51	**15.1**	0.6	12.3
`rnd#11`	114	438.31	197.32	471.63	182.44	**7.9**	−0.7	7.3
`rnd#12`	110	73.44	20.98	53.08	13.16	**23.8**	2.3	7.5
`rnd#13`	123	24.86	14.33	13.22	11.37	7.1	2.9	**10.9**
`rnd#14`	124	156.53	89.51	84.94	90.49	4.8	5.1	**8.4**
`rnd#15`	116	654.23	320.22	521.43	289.63	**7.3**	2.9	6.2
`rnd#16`	109	5,259.73	2,170.50	1,271.24	1,342.31	**11.7**	4.7	10.6
`rnd#17`	111	3,870.85	1,096.95	944.77	1,126.95	9.0	5.8	**14.2**
`rnd#18`	110	1,085.21	216.91	602.61	376.10	10.0	4.4	**10.9**
`rnd#19`	110	10,496.40	1,971.21	3,847.48	708.52	9.0	**15.8**	10.4
`rnd#20`	114	540.62	90.82	382.95	359.84	**10.4**	3.6	4.2

similar performance, the HA family (hole and anti-hole inequalities) generated the highest per-inequality improvement on 9 instances. On the other hand, the SD family (star and double-star inequalities) showed an inferior performance compared to the two former families. In fact, on two instances `rnd#3` and `rnd#11`, incorporating these inequalities into the base formulation even resulted in a slight increase in the solution time. This observation may be justified by noting that star and double-star subgraphs generate exponentially many (with respect to the size of the subgraph) known facet-defining valid inequalities for the IUC polytope associated with the supergraph, whereas this number for the other two families is much smaller, and we have incorporated only a couple of these inequalities from each family.

In the next phase of our experiments, we also examined the effect of adding different combinations of these valid inequalities on the solution time of the test instances. Table 4.3 presents the corresponding results. In this table, "+HA&SD" shows the solution time when the base formulation has been strengthened by the combination of HA and SD families. Similarly, "+HA&FW" and "+SD&FW" show the results for the combination of HA and FW families, and SD and FW families, respectively. The last column of this table, i.e., "+All", presents the solution time for each instance obtained by incorporating all three HA, SD, and FW families in the base formulation (4.5). Note that the total number of appended inequalities in this case is about 1% of the number of OT inequalities for each instance. The boldface numbers in this table show the best solution time obtained for each instance. Even by incorporating a rather small number of valid inequalities, the solution times are substantially better than those obtained due to incorporating individual families of the valid inequalities, presented in Table 4.2. Interestingly, for all instances, the best solution time was attained when the FW family was incorporated, and only on 11 instances (out of 20), the best solution time was due to incorporating all valid inequalities.

The results of our first set of experiments reveal the merit of the proposed valid inequalities. In practice, however, the structure of an input graph is not known *a priori*. Therefore, integer (linear) programming solution methods of the maximum IUC problem should rely on subgraph-detection routines in order to generate these valid inequalities. In our second set of experiments, we investigated efficacy of the proposed valid inequalities under this setting. We generated a set of (Erdös-Rényi) random graphs $G(n,p)$ with $n = 100$ and p varying from 0.05 to 0.95. In these graphs, n is the number of vertices and an edge exists between every pair of vertices with a probability p independent of the others. The characteristics of our second set of test instances, including the number of open triangles $|\Lambda|$ for each instance, are shown in Table 4.4. In this table, the numerical part of the name of each instance shows the corresponding value of p, which is also the expected edge density of the graph. In light of the results of our first set of experiments, as well as the complexity results of the previous section, we just considered 4-hole, wheel and fan cutting planes in this part of our experiments, and applied them all to the root node of the branch-and-cut tree. It should be mentioned that developing a full-blown branch-and-cut algorithm for

Table 4.3: First set of experiments: results of incorporating different combinations of valid inequalities.

	Solution time (CPU sec.)				
Name	Base	+HA&SD	+HA&FW	+SD&FW	+All
`rnd#1`	3,094.30	477.81	345.94	291.17	**167.73**
`rnd#2`	1,103.08	923.40	637.46	**237.37**	354.18
`rnd#3`	90.77	29.96	19.81	25.13	**16.91**
`rnd#4`	197.38	158.29	103.24	**91.26**	103.08
`rnd#5`	1,218.80	326.88	356.16	583.89	**242.28**
`rnd#6`	9,820.76	274.15	**119.29**	1,346.66	157.43
`rnd#7`	173.19	32.46	36.02	58.75	**23.43**
`rnd#8`	115.95	26.17	**19.85**	84.22	25.78
`rnd#9`	95.45	68.12	17.80	19.70	**15.05**
`rnd#10`	48.18	11.85	15.05	9.29	**7.47**
`rnd#11`	438.31	152.38	**123.44**	158.43	131.15
`rnd#12`	73.44	28.09	**10.14**	14.47	14.18
`rnd#13`	24.86	7.28	**1.99**	8.93	8.40
`rnd#14`	156.53	97.76	50.99	87.31	**47.58**
`rnd#15`	654.23	287.86	147.35	216.37	**103.77**
`rnd#16`	5,259.73	722.94	665.09	414.69	**157.23**
`rnd#17`	3,870.85	846.85	635.40	510.96	**332.18**
`rnd#18`	1,085.21	138.07	127.95	223.97	**102.35**
`rnd#19`	10,496.40	1,391.08	**283.62**	546.51	354.59
`rnd#20`	540.62	93.92	**61.29**	191.92	68.03

the maximum IUC problem is out of the scope this paper, and the purpose of our experiments was to test effectiveness of employing these inequalities using a straightforward strategy, which is described below.

For each instance, we identified the entire set of 4-holes, but we employed a subset of them to generate the 4-hole cutting planes. Note that the expected number of 4-holes in random graphs with moderate densities is much higher than the expected number of open triangles. Besides, every 4-hole inequality (4.7) dominates four OT inequalities, which can be eliminated from the original formulation upon adding the 4-hole cuts. In this regard, our solution strategy was to find a small subset of 4-hole inequalities that could cover a large portion (possibly all) of OT inequalities. Clearly, finding a minimum number of 4-holes to dominate all possible open triangles

Table 4.4: Second set of experiments: cut-generation characteristics.

				# Valid inequalities (cuts)				Cut generation
Name	$\|V\|$	$\|E\|$	$\|\Lambda\|$	OT	4-hole	Wheel	Fan	(CPU sec.)
RND_0.05	100	255	1,204	956	70	0	0	0.05
RND_0.10	100	503	4,558	1,867	868	9	0	0.06
RND_0.15	100	810	10,823	1,235	3,444	378	211	0.34
RND_0.20	100	1,058	17,337	511	6,126	1,614	965	1.40
RND_0.25	100	1,261	23,406	274	8,378	2,463	1,323	3.57
RND_0.30	100	1,541	32,444	30	10,945	3,338	1,466	9.623
RND_0.35	100	1,812	41,242	7	13,683	3,382	1,616	20.19
RND_0.40	100	1,966	45,987	2	14,988	3,470	1,528	27.58
RND_0.45	100	2,253	54,553	1	17,523	3,506	1,494	46.07
RND_0.50	100	2,470	60,337	0	19,323	3,456	1,544	63.04
RND_0.55	100	2,773	67,027	0	21,103	3,604	1,396	87.25
RND_0.60	100	2,981	70,308	0	21,897	3,687	1,313	101.20
RND_0.65	100	3,223	71,881	0	22,766	3,589	1,384	108.66
RND_0.70	100	3,457	71,135	0	22,923	3,747	1,184	104.21
RND_0.75	100	3,748	67,644	0	21,707	2,328	700	91.00
RND_0.80	100	3,994	61,000	0	19,990	350	62	66.70
RND_0.85	100	4,259	50,326	0	16,822	2	24	38.02
RND_0.90	100	4,460	39,011	2	13,635	0	0	17.86
RND_0.95	100	4,726	19,977	762	7,509	0	0	2.53

in a graph is a set covering problem, which is NP-hard in general. We employed a well-known greedy heuristic for this problem to generate the 4-hole cuts; at every iteration, we selected a 4-hole with the maximum number of uncovered open triangles, and stopped when this number was zero for all the remaining 4-holes. Obviously, not all open triangles are guaranteed to be part of a 4-hole in a graph, thus not all OT inequalities could be replaced with the generated 4-hole cuts. Table 4.4 shows the number of generated 4-hole cuts for each instance, as well as the number of OT inequalities that were not covered by them and remained in the problem formulation.

We also generated a set of wheel and fan cuts for each instance. To do this, for every vertex $i \in V$, we considered the subgraph induced by its neighbors, i.e., $G[N(i)]$, and detected a set of chordless cycles and paths using a simple enumeration method. Clearly, a graph may contain an exponential number of such structures, thus we restricted the search on each subgraph to at most

50 cycles and paths or 1.00 CPU second, whichever came first before the search was complete. In order to generate high-quality wheel and fan cuts, we only considered (chordless) cycles and paths with at least 7 vertices that satisfied the facet-defining conditions of Theorems 21 and 23. The number of generated wheel and fan cuts for each instance is presented in Table 4.4. The last column of this table shows the aggregate time spent on generating all 4-hole, wheel and fan cuts for each instance. It should be mentioned that, for all instances, the CPU time taken to generate the wheel and fan inequalities was negligible compared to the required time to generate the 4-hole cuts. In fact, the maximum time spent by the algorithm to generate the entire set of wheel and fan cuts for an instance remained below 1.20 CPU seconds.

The results of the second set of experiments are presented in Table 4.5. We solved each instance using the base formulation (4.5), as well as its enhancement obtained by adding the generated valid inequalities, denoted by "+VI" in Table 4.5. The second column of this table shows the optimal solution value of the maximum IUC problem for each instance. The next two columns present the optimal solution value of the corresponding LP relaxation problems. The last four columns of this table compare the size of the branch-and-cut (B&C) trees and solution times. In this set of experiments, the automatic cut-generation of the solver was also on. The results clearly show the effectiveness of the proposed valid inequalities. Significant improvements in the solution times are observed for the graphs with the edge density of less than 65%, as well as those with the edge density of 80% and more, due to incorporating the generated cutting planes. The +VI solution times were slightly better than those for the base formulation on `RND_0.65` and `RND_0.75`, and slightly worse on `RND_0.70`. The better solution time for each instance is shown in bold in Table 4.5. For all instances, the number of B&C nodes was considerably smaller under the +VI strategy.

Table 4.5: Second set of experiments: integer (linear) programming solution results.

		LP rlx. opt. value		# B&C nodes		Solution time (CPU sec.)	
Name	opt.	Base	+VI	Base	+VI	Base	+VI
RND_0.05	53	66.67	59.22	3,241	0	22.51	**5.63**
RND_0.10	39	66.67	49.71	213,358	17,005	1,778.93	**218.46**
RND_0.15	28	66.67	46.60	2,017,952	160,544	35,357.90	**2,284.37**
RND_0.20	24	66.67	44.92	1,112,661	180,755	23,173.20	**4,248.66**
RND_0.25	21	66.67	44.30	1,159,561	192,961	34,036.80	**5,123.73**
RND_0.30	18	66.67	44.29	724,553	227,952	35,861.80	**8,733.08**
RND_0.35	16	66.67	44.33	615,306	191,929	30,004.00	**8,444.93**
RND_0.40	15	66.67	44.44	468,742	125,469	26,178.80	**6,928.29**
RND_0.45	13	66.67	44.44	295,550	126,449	19,691.90	**7,250.91**
RND_0.50	12	66.67	44.44	222,235	104,277	18,260.00	**8,861.15**
RND_0.55	11	66.67	44.44	176,862	70,025	13,810.80	**6,621.82**
RND_0.60	11	66.67	44.44	111,914	49,889	9,102.53	**5,930.36**
RND_0.65	13	66.67	44.44	66,995	33,969	5,615.65	**4,927.96**
RND_0.70	15	66.67	44.44	57,423	31,844	**5,273.13**	5,414.28
RND_0.75	19	66.67	44.48	31,135	19,714	5,163.23	**4,490.54**
RND_0.80	21	66.67	46.63	74,719	29,461	7,943.90	**4,605.04**
RND_0.85	25	66.67	49.75	46,781	22,872	4,574.90	**2,404.52**
RND_0.90	31	66.67	50.00	18,562	7,787	1,897.80	**492.21**
RND_0.95	46	66.67	51.50	0	0	34.24	**5.16**

5. SUMMARY AND CONCLUSIONS

In this dissertation, we addressed two optimization problems concerning cluster analysis of graphs: the maximum edge weight clique and maximum induced cluster subgraph problems.

In Chapter 2, we introduced a nonconvex quadratic-programming formulation for the maximum edge weight clique (MEWC) problem. Optimality characteristics of the new formulation were studied, and the correspondence between local and global optima of the continuous problem and spacial structures in the underlying graph was established. Mainly, it was shown that a feasible solution is a local maximizer of the continuous problem if and only if it is the characteristic vector of a maximal clique in the graph. Consequently, a global maximizer characterizes a maximum edge weight clique.

Based on the continuous formulation and its optimality characterization, a new exact algorithm for the MEWC problem was presented. The algorithm is a combinatorial branch-and-bound procedure that takes advantage of an algebraic upper bound, which is derived from a quadratic relaxation of the continuous problem. The branch-and-bound algorithm also uses an initial lower bound that is provided by a construction heuristic method. The heuristic algorithm extracts a maximal clique based on solving an approximation of the continuous problem. We conducted computational experiments on 28 benchmark instances, and compared the solution time of our algorithm with the best results on the same instances in the literature. The experiments show significant improvements; in particular, we could solve 24 instances within the time limit of 3 hours (for each instance). We also used the computational results to show the quality of the presented algebraic bound explicitly. Our results indicate the effectiveness of our method, especially on relatively dense graphs.

In future research, one may devise other heuristic algorithms for the MEWC problem by different approximations of the continuous problem presented in this dissertation. Besides, other properties of the presented formulation may be explored to find tighter algebraic upper bounds for this problem.

In Chapter 3, we used a Lagrangian relaxation of an integer (linear) programming formulation of the MEWC problem to derive a new analytic (closed-form) upper bound on the clique number of a graph. The presented bound is characterized by the degrees of the vertices in the graph; hence, it is computable in linear time. We compared this bound with other analytic bounds proposed in the literature. Our computational results showed that the Lagrangian relaxation bound is in general tighter than the bound introduced by Amin and Hakimi [58], which was the only existing analytic upper bound on the clique number computable in linear time.

We also presented a new exact algorithm for the MEWC problem. The algorithm is a combinatorial branch-and-bound procedure that employs an efficient combinatorial pruning method. At every node of the search tree, the algorithm uses an upper bound on the clique number of the corresponding subgraph to calculate an upper bound for the MEWC problem. The calculation of this bound is based on contribution of the vertices of a clique to its weight, rather than the edges of the induced subgraph. We presented computational results of our algorithm on some benchmark instances, and compared them with the existing results in the literature. It was shown that the proposed algorithm outperforms the preceding solution methods of the MEWC problem due to effectiveness of its pruning process. Our solution times are at least an order of magnitude better than the best reported results for all nontrivial instances considered in previous works for this problem, including the algorithm presented in Chapter 2 of this dissertation. The computational results also showed that recoloring approaches and sequential elimination enhancements to draw tight upper bounds on the clique number are not as effective as in the maximum clique problem, due to the overhead of calculating the pruning threshold and enhanced bounds for the MEWC problem.

Developing better coloring-based upper bounds on the clique number that exploit the structure of the colorings used may significantly speed up the proposed B&B algorithm. In addition, incorporating fast and effective heuristics could improve the performance of the proposed approach, especially on dense graphs with large clique number, for which the classical maximum clique algorithms tend to perform well.

In Chapter 4, we presented a study of the IUC polytope associated with a simple undirected graph, defined as the convex hull of the incidence vectors of all its vertex subsets inducing cluster subgraphs. It was shown that the fractional IUC polytope, obtained from relaxing integrality of the variables in the definition of the original polytope, is a polyhedral outer-approximation of a cubically-constrained region in the space of the original variables. Hence, it provides a very weak approximation of the (integral) IUC polytope, remarking the importance of incorporating strong valid inequalities in integer (linear) programming solution methods of the maximum IUC problem. We derived several facet-defining valid inequalities for the IUC polytope associated with cycle, anti-cycle, star, double-star, fan and wheel graphs, along with the conditions that they remain facet-defining for the IUC polytope associated with a supergraph containing them, as well as some results concerning the corresponding lifting procedures. We also presented a complete description of the IUC polytope for some classes of these graphs. In our presentation, we spotted the similarities between the facial structure of the IUC polytope and those of the independent set (vertex packing) and clique polytopes. For each family of the proposed valid inequalities, we studied the computational complexity of the corresponding separation problem. In particular, we showed that the separation problem for the families of star, double-star, fan and wheel inequalities are NP-hard. We also examined the effectiveness of the proposed valid inequalities when employed in an integer (linear) programming solution method of the maximum IUC problem through computational experiments. Our results showed great improvement in solution time of this problem, revealing the merit of the proposed valid inequalities from the computational standpoint.

As an extension of this work, one may search for other facet-producing structures for the IUC polytope. An immediate step could be to investigate webs that subsume cycle and anti-cycle graphs. Developing detailed branch-and-cut algorithms for the maximum IUC problem based on effective heuristic separation procedures for the proposed valid inequalities in this paper is another direction for future research. Another interesting direction is to explore LP-based scale reduction techniques for the maximum IUC problem. It should be mentioned that, because of the cubic nature of the IUC formulation, the fractional IUC polytope does not generally hold the favorable structural properties of the fractional independent set (vertex packing) and clique polytopes such as persistency [125]. However, we observed through a set of computational experiments that the frequency of non-persistent optimal solutions to the corresponding LP relaxation problem is not high. In this regard, it is of interest to identify the conditions under which an extreme point of the fractional IUC polytope shows persistency. Finally, we note that several other integer (linear) programming formulations for the maximum IUC problem can be obtained from the cubic formulation presented in this dissertation, through different linearization techniques, such as the method of Sherali and Adams [126] or the one of Boros and Hammer [127] in higher-dimensional spaces than the space of the original variables. These formulations will have theoretically stronger LP relaxations than its natural integer programming formulation; hence, it is also interesting to compare their performances in practice.

REFERENCES

[1] R. Luce and A. Perry, “A method of matrix analysis of group structure,” *Psychometrika*, vol. 14, pp. 95–116, 1949.

[2] S. Hosseinian, D. B. M. M. Fontes, and S. Butenko, “A nonconvex quadratic optimization approach to the maximum edge weight clique problem,” *Journal of Global Optimization*, vol. 72, no. 2, pp. 219–240, 2018.

[3] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*, pp. 85–103, Plenum Press, New York, 1972.

[4] M. Pavan and M. Pelillo, “Generalizing the motzkin-straus theorem to edge-weighted graphs, with applications to image segmentation,” in *Energy Minimization Methods in Computer Vision and Pattern Recognition: 4th International Workshop, EMMCVPR 2003, Lisbon, Portugal, July 7-9, 2003. Proceedings* (A. Rangarajan, M. Figueiredo, and J. Zerubia, eds.), pp. 485–500, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[5] T. Ma and L. J. Latecki, “Maximum weight cliques with mutex constraints for video object segmentation,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 670–677, June 2012.

[6] L. Cavique, “A scalable algorithm for the market basket analysis,” *Journal of Retailing and Consumer Services*, vol. 14, no. 6, pp. 400–407, 2007.

[7] T. Akutsu, M. Hayashida, E. Tomita, and J. Suzuki, “Protein threading with profiles and constraints,” in *Proceedings. Fourth IEEE Symposium on Bioinformatics and Bioengineering*, pp. 537–544, May 2004.

[8] J. B. Brown, D. B. K. C., E. Tomita, and T. Akutsu, “Multiple methods for protein side chain packing using maximum weight cliques,” *Genome Informatics*, vol. 17, pp. 3–12, 2006.

[9] M. A. Jabbar, B. L. Deekshatulu, and P. Chandra, “Graph based approach for heart disease prediction,” in *Proceedings of the Third International Conference on Trends in Information, Telecommunication and Computing* (V. V. Das, ed.), pp. 465–474, New York, NY: Springer New York, 2013.

[10] G. Dijkhuizen and U. Faigle, “A cutting-plane approach to the edge-weighted maximal clique problem,” *European Journal of Operational Research*, vol. 69, no. 1, pp. 121–130, 1993.

[11] K. Park, K. Lee, and S. Park, “An extended formulation approach to the edge-weighted maximal clique problem,” *European Journal of Operational Research*, vol. 95, no. 3, pp. 671–682, 1996.

[12] E. M. Macambira and C. C. de Souza, “The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations,” *European Journal of Operational Research*, vol. 123, no. 2, pp. 346–371, 2000.

[13] M. M. Sorensen, “New facets and a branch-and-cut algorithm for the weighted clique problem,” *European Journal of Operational Research*, vol. 154, no. 1, pp. 57–70, 2004.

[14] M. Hunting, U. Faigle, and W. Kern, “A Lagrangian relaxation approach to the edge-weighted clique problem,” *European Journal of Operational Research*, vol. 131, no. 1, pp. 119–131, 2001.

[15] B. Alidaee, F. Glover, G. Kochenberger, and H. Wang, “Solving the maximum edge weight clique problem via unconstrained quadratic programming,” *European Journal of Operational Research*, vol. 181, no. 2, pp. 592–597, 2007.

[16] G. Palubeckis, “Iterated tabu search for the maximum diversity problem,” *Applied Mathematics and Computation*, vol. 189, no. 1, pp. 371–383, 2007.

[17] R. Aringhieri and R. Cordone, “Comparing local search metaheuristics for the maximum diversity problem,” *Journal of the Operational Research Society*, vol. 62, no. 2, pp. 266–280, 2011.

[18] Y. Wang, J. K. Hao, F. Glover, and Z. Lü, "A tabu search based memetic algorithm for the maximum diversity problem," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 103–114, 2014.

[19] M. Gallego, A. Duarte, M. Laguna, and R. Martí, "Hybrid heuristics for the maximum diversity problem," *Computational Optimization and Applications*, vol. 44, no. 3, pp. 411–426, 2009.

[20] M. R. Q. de Andrade, P. M. F. de Andrade, S. L. Martins, and A. Plastino, "Grasp with path-relinking for the maximum diversity problem," in *Experimental and Efficient Algorithms: 4th International Workshop, WEA 2005, Santorini Island, Greece, May 10-13, 2005. Proceedings* (S. E. Nikoletseas, ed.), pp. 558–569, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[21] G. C. Silva, M. R. Q. de Andrade, L. S. Ochi, S. L. Martins, and A. Plastino, "New heuristics for the maximum diversity problem," *Journal of Heuristics*, vol. 13, no. 4, pp. 315–336, 2007.

[22] R. Martí, M. Gallego, A. Duarte, and E. G. Pardo, "Heuristics and metaheuristics for the maximum diversity problem," *Journal of Heuristics*, vol. 19, no. 4, pp. 591–615, 2013.

[23] Q. Wu and J.-K. Hao, "A review on algorithms for maximum clique problems," *European Journal of Operational Research*, vol. 242, no. 3, pp. 693–709, 2015.

[24] W. Pullan, "Approximating the maximum vertex/edge weighted clique using local search," *Journal of Heuristics*, vol. 14, no. 2, pp. 117–134, 2008.

[25] L. Gouveia and P. Martins, "Solving the maximum edge-weight clique problem in sparse graphs with compact formulations," *EURO Journal on Computational Optimization*, vol. 3, no. 1, pp. 1–30, 2015.

[26] S. Hosseinian, D. B. M. M. Fontes, and S. Butenko, "A quadratic approach to the maximum edge weight clique problem.," in *XIII Global Optimization Workshop (GOW'16)* (A. M.

A. C. Rocha, M. F. P. Costa, and E. M. G. P. Fernandes, eds.), pp. 125–128, Braga, Portugal: University of Minho, 2016.

[27] S. Hosseinian, D. B. M. M. Fontes, and S. Butenko, "The maximum edge weight clique problem: Formulations and solution approaches," in *Optimization Methods and Applications* (S. Butenko, P. M. Pardalos, and V. Shylo, eds.), pp. 209–227, Springer, 2017.

[28] T. S. Motzkin and E. G. Straus, "Maxima for graphs and a new proof of a theorem of Turán," *Canad. J. Math.*, vol. 17, pp. 533–540, 1965.

[29] P. Pardalos and A. Phillips, "A global optimization approach for solving the maximum clique problem," *Int. J. Comput. Math.*, vol. 33, pp. 209–216, 1990.

[30] M. Pelillo and A. Jagota, "Feasible and infeasible maxima in a quadratic program for maximum clique," *Journal of Artificial Neural Networks*, vol. 2, pp. 411–420, 1995.

[31] L. E. Gibbons, D. W. Hearn, and P. M. Pardalos, "A continuous based heuristic for the maximum clique problem," in *Cliques, Coloring, and Satisfiability: Second DIMACS Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 26* (D. S. Johnson and M. A. Trick, eds.), pp. 103–124, Providence, RI: American Mathematical Society, 1996.

[32] L. E. Gibbons, D. W. Hearn, P. M. Pardalos, and M. V. Ramana, "Continuous characterizations of the maximum clique problem," *Mathematics of Operations Research*, vol. 22, pp. 754–768, 1997.

[33] I. M. Bomze, "Evolution towards the maximum clique," *Journal of Global Optimization*, vol. 10, pp. 143–164, 1997.

[34] I. M. Bomze, M. Budinich, M. Pelillo, and C. Rossi, "A new "annealed" heuristic for the maximum clique problem," in *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems* (P. M. Pardalos, ed.), pp. 78–96, Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000.

[35] S. R. Bulò and M. Pelillo, "A generalization of the Motzkin–Straus theorem to hypergraphs," *Optimization Letters*, vol. 3, no. 2, pp. 287–295, 2009.

[36] S. Busygin, "A new trust region technique for the maximum weight clique problem," *Discrete Applied Mathematics*, vol. 154, pp. 2080–2096, 2006.

[37] Y. Peng, H. Peng, Q. Tang, and C. Zhao, "An extension of the Motzkin–Straus theorem to non-uniform hypergraphs and its applications," *Discrete Applied Mathematics*, vol. 200, pp. 170–175, 2016.

[38] J. Harant, "A lower bound on the independence number of a graph," *Discrete Mathematics*, vol. 188, pp. 239–243, 1998.

[39] J. Harant, A. Pruchnewski, and M. Voigt, "On dominating sets and independent sets of graphs," *Combinatorics, Probability and Computing*, vol. 8, pp. 547–553, 1999.

[40] J. Harant, "Some news about the independence number of a graph," *Discussiones Mathematicae Graph Theory*, vol. 20, pp. 71–79, 2000.

[41] J. Abello, S. Butenko, P. Pardalos, and M. Resende, "Finding independent sets in a graph using continuous multivariable polynomial formulations," *Journal of Global Optimization*, vol. 21, pp. 111–137, 2001.

[42] S. Busygin, S. Butenko, and P. M. Pardalos, "A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere," *Journal of Combinatorial Optimization*, vol. 6, pp. 287–297, 2002.

[43] B. Balasundaram and S. Butenko, "On a polynomial fractional formulation for independence number of a graph," *Journal of Global Optimization*, vol. 35, pp. 405–421, 2006.

[44] P. M. Pardalos and S. A. Vavasis, "Quadratic programming with one negative eigenvalue is NP-hard," *Journal of Global Optimization*, vol. 1, no. 1, pp. 15–22, 1991.

[45] Y. Ye, "A new complexity result on minimization of a quadratic function with a sphere constraint," in *Recent Advances in Global Optimization*, pp. 19–31, Princeton University Press, 1992.

[46] G. E. Forsythe and G. H. Golub, "On the stationary values of a second-degree polynomial on the unit sphere," *Journal of the Society for Industrial and Applied Mathematics*, vol. 13, no. 4, pp. 1050–1068, 1965.

[47] R. Carraghan and P. Pardalos, "An exact algorithm for the maximum clique problem," *Operations Research Letters*, vol. 9, pp. 375–382, 1990.

[48] P. R. J. Östergård, "A fast algorithm for the maximum clique problem," *Discrete Applied Mathematics*, vol. 120, pp. 197–207, 2002.

[49] E. Tomita and T. Kameda, "An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments," *Journal of Global Optimization*, vol. 37, no. 1, pp. 95–111, 2007.

[50] E. Tomita, Y. Sutani, T. Higashi, S. Takahashi, and M. Wakatsuki, "A simple and faster branch-and-bound algorithm for finding a maximum clique," in *WALCOM: Algorithms and Computation: 4th International Workshop, WALCOM 2010, Dhaka, Bangladesh, February 10-12, 2010. Proceedings* (M. S. Rahman and S. Fujita, eds.), pp. 191–203, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[51] M. Batsyn, B. Goldengorin, E. Maslov, and P. M. Pardalos, "Improvements to MCS algorithm for the maximum clique problem," *Journal of Combinatorial Optimization*, vol. 27, no. 2, pp. 397–416, 2014.

[52] P. San Segundo, A. Nikolaev, and M. Batsyn, "Infra-chromatic bound for exact maximum clique search," *Computers & Operations Research*, vol. 64, pp. 293–303, 2015.

[53] D. S. Johnson and M. A. Trick, eds., *Cliques, Coloring, and Satisfiability: Second* DIMACS *Implementation Challenge*. Providence, RI: American Mathematical Society, 1996.

[54] S. Hosseinian, D. B. M. M. Fontes, and S. Butenko, “A lagrangian bound on the clique number and an exact algorithm for the maximum edge weight clique problem,” *INFORMS Journal on Computing*, vol. 32, no. 3, pp. 747–762, 2020.

[55] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, “The maximum clique problem,” in *Handbook of Combinatorial Optimization* (D.-Z. Du and P. M. Pardalos, eds.), (Dordrecht, The Netherlands), pp. 1–74, Kluwer Academic Publishers, 1999.

[56] D. Zuckerman, “Linear degree extractors and the inapproximability of max clique and chromatic number,” in *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, (New York), pp. 681–690, ACM, 2006.

[57] H. S. Wilf, “The eigenvalues of a graph and its chromatic number,” *J. London Math. Soc.*, vol. 42, pp. 330–332, 1967.

[58] A. T. Amin and S. L. Hakimi, “Upper bounds on the order of a clique of a graph,” *SIAM J. Appl. Math.*, vol. 22, pp. 569–573, 1972.

[59] M. Budinich, “Exact bounds on the order of the maximum clique of a graph,” *Discrete Applied Mathematics*, vol. 127, pp. 535–543, 2003.

[60] M. Padberg, “The boolean quadric polytope: some characteristics, facets and relatives,” *Mathematical Programming*, vol. 45, no. 1-3, pp. 139–172, 1989.

[61] A. Mehrotra and M. A. Trick, “Cliques and clustering: A combinatorial approach,” *Operations Research Letters*, vol. 22, no. 1, pp. 1–12, 1998.

[62] R. Martí, M. Gallego, and A. Duarte, “A branch and bound algorithm for the maximum diversity problem,” *European Journal of Operational Research*, vol. 200, no. 1, pp. 36–44, 2010.

[63] D. B. M. M. Fontes, J. F. Gonçalves, and F. A. C. C. Fontes, “An evolutionary approach to the maximum edge weight clique problem,” *Recent Advances in Electrical & Electronic Engineering*, 2018.

[64] W. Pullan, “Phased local search for the maximum clique problem,” *Journal of Combinatorial Optimization*, vol. 12, no. 3, pp. 303–323, 2006.

[65] L. Lovász, “On the shannon capacity of a graph,” *IEEE Trans. Inform. Theory*, vol. 25, pp. 1–7, 1979.

[66] S. R. Bulò and M. Pelillo, “A new spectral bound on the clique number of graphs,” in *Structural, Syntactic, and Statistical Pattern Recognition. SSPR /SPR 2010.* (E. Hancock, R. Wilson, T. Windeatt, I. Ulusoy, and F. Escolano, eds.), vol. 6218 of *Lecture Notes in Computer Science*, pp. 680–689, Berlin, Heidelberg: Springer, 2010.

[67] J. Mycielski, “Sur le coloriage des graphs,” *Colloquium Mathematicae*, vol. 3, no. 2, pp. 161–162, 1955.

[68] E. Tomita and T. Seki, “An efficient branch-and-bound algorithm for finding a maximum clique,” in *Discrete Mathematics and Theoretical Computer Science: 4th International Conference, DMTCS 2003 Dijon, France, July 7–12, 2003 Proceedings* (C. S. Calude, M. J. Dinneen, and V. Vajnovszki, eds.), pp. 278–289, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[69] C. M. Li and Z. Quan, “An efficient branch-and-bound algorithm based on MaxSAT for the maximum clique problem,” in *AAAI*, vol. 10, pp. 128–133, 2010.

[70] P. San Segundo, D. Rodrııguez-Losada, and A. Jiménez, “An exact bit-parallel algorithm for the maximum clique problem,” *Computers & Operations Research*, vol. 38, pp. 571–581, 2011.

[71] B. Gendron, A. Hertz, and P. St-Louis, “A sequential elimination algorithm for computing bounds on the clique number of a graph,” *Discrete Optimization*, vol. 5, pp. 615–628, 2008.

[72] S. Shimizu, K. Yamaguchi, and S. Masuda, “A branch-and bound based exact algorithm for the maximum edge-weight clique problem,” in *Computational Science/Intelligence & Applied Informatics* (R. Lee, ed.), vol. 787 of *Studies in Computational Intelligence*, pp. 27–47, Springer International Publishing AG, 2019.

[73] S. Hosseinian and S. Butenko, "Polyhedral properties of the induced cluster subgraphs," *Discrete Applied Mathematics*, vol. 297, pp. 80–96, 2021.

[74] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

[75] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Berlin: Springer-Verlag, 2nd ed., 1993.

[76] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York: Wiley, 1999.

[77] J. Pattillo, N. Youssef, and S. Butenko, "On clique relaxation models in network analysis," *European Journal of Operational Research*, vol. 226, no. 1, pp. 9–18, 2013.

[78] F. V. Fomin, S. Gaspers, D. Kratsch, M. Liedloff, and S. Saurabh, "Iterative compression and exact algorithms," *Theoretical Computer Science*, vol. 411, no. 7-9, pp. 1045–1053, 2010.

[79] Z. Ertem, A. Veremyev, and S. Butenko, "Detecting large cohesive subgroups with high clustering coefficients in social networks," *Social Networks*, vol. 46, pp. 1–10, 2016.

[80] B. Balasundaram and S. Butenko, "On a polynomial fractional formulation for independence number of a graph," *Journal of Global Optimization*, vol. 35, no. 3, pp. 405–421, 2006.

[81] Z. Ertem, E. Lykhovyd, Y. Wang, and S. Butenko, "The maximum independent union of cliques problem: complexity and exact approaches," *Journal of Global Optimization*, vol. 76, no. 3, pp. 545–562, 2020.

[82] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier, "Automated generation of search tree algorithms for hard graph modification problems," *Algorithmica*, vol. 39, no. 4, pp. 321–347, 2004.

[83] J. M. Lewis and M. Yannakakis, "The node-deletion problem for hereditary properties is NP-complete," *Journal of Computer and System Sciences*, vol. 20, no. 2, pp. 219–230, 1980.

[84] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier, "Fixed-parameter algorithms for cluster vertex deletion," *Theory of Computing Systems*, vol. 47, no. 1, pp. 196–217, 2010.

[85] A. Boral, M. Cygan, T. Kociumaka, and M. Pilipczuk, "A fast branching algorithm for cluster vertex deletion," *Theory of Computing Systems*, vol. 58, no. 2, pp. 357–376, 2016.

[86] T.-N. Le, D. Lokshtanov, S. Saurabh, S. Thomassé, and M. Zehavi, "Subquadratic kernels for implicit 3-hitting set and 3-set packing problems," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 331–342, SIAM, 2018.

[87] F. V. Fomin, S. Gaspers, D. Lokshtanov, and S. Saurabh, "Exact algorithms via monotone local search," in *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, (New York, NY), pp. 764–775, ACM, 2016.

[88] J. You, J. Wang, and Y. Cao, "Approximate association via dissociation," *Discrete Applied Mathematics*, vol. 219, pp. 202–209, 2017.

[89] S. Fiorini, G. Joret, and O. Schaudt, "Improved approximation algorithms for hitting 3-vertex paths," in *Integer Programming and Combinatorial Optimization. IPCO 2016* (Q. Louveaux and M. Skutella, eds.), vol. 9682 of *Lecture Notes in Computer Science*, (Cham), pp. 238–249, Springer, 2016.

[90] R. Shamir, R. Sharan, and D. Tsur, "Cluster graph modification problems," *Discrete Applied Mathematics*, vol. 144, no. 1-2, pp. 173–182, 2004.

[91] F. Dehne, M. A. Langston, X. Luo, S. Pitre, P. Shaw, and Y. Zhang, "The cluster editing problem: Implementations and experiments," in *Parameterized and Exact Computation. IWPEC 2006* (H. L. Bodlaender and M. A. Langston, eds.), vol. 4169 of *Lecture Notes in Computer Science*, (Berlin, Heidelberg), pp. 13–24, Springer, 2006.

[92] S. Böcker, "A golden ratio parameterized algorithm for cluster editing," *Journal of Discrete Algorithms*, vol. 16, pp. 79–89, 2012.

[93] C. Komusiewicz and J. Uhlmann, “Cluster editing with locally bounded modifications,” *Discrete Applied Mathematics*, vol. 160, no. 15, pp. 2259–2270, 2012.

[94] S. Böcker and J. Baumbach, “Cluster editing,” in *The Nature of Computation. Logic, Algorithms, Applications. CiE 2013* (P. Bonizzoni, V. Brattka, and B. Löwe, eds.), vol. 7921 of *Lecture Notes in Computer Science*, pp. 33–44, Berlin, Heidelberg: Springer, 2013.

[95] F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger, “Tight bounds for parameterized complexity of cluster editing with a small number of clusters,” *Journal of Computer and System Sciences*, vol. 80, no. 7, pp. 1430–1447, 2014.

[96] L. Bastos, L. S. Ochi, F. Protti, A. Subramanian, I. C. Martins, and R. G. S. Pinheiro, “Efficient algorithms for cluster editing,” *Journal of Combinatorial Optimization*, vol. 31, no. 1, pp. 347–371, 2016.

[97] K. Jansen, P. Scheffler, and G. Woeginger, “The disjoint cliques problem,” *RAIRO-Operations Research*, vol. 31, no. 1, pp. 45–66, 1997.

[98] B. P. W. Ames and S. A. Vavasis, “Convex optimization for the planted k-disjoint-clique problem,” *Mathematical Programming*, vol. 143, pp. 299–337, Feb 2014.

[99] R. Van Bevern, H. Moser, and R. Niedermeier, “Approximation and tidying – a problem kernel for s-plex cluster vertex deletion,” *Algorithmica*, vol. 62, no. 3-4, pp. 930–950, 2012.

[100] J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann, “A more relaxed model for graph-based data clustering: s-plex cluster editing,” *SIAM Journal on Discrete Mathematics*, vol. 24, no. 4, pp. 1662–1683, 2010.

[101] M. S. Krishnamoorthy and N. Deo, “Node-deletion NP-complete problems,” *SIAM Journal on Computing*, vol. 8, no. 4, pp. 619–625, 1979.

[102] M. Pilipczuk, “Exact algorithms for induced subgraph problems,” in *Encyclopedia of Algorithms* (M.-Y. Kao, ed.), pp. 674–678, New York, NY: Springer New York, 2016.

[103] M. W. Padberg, "A note on zero-one programming," *Operations Research*, vol. 23, pp. 833–837, 1975.

[104] L. A. Wolsey, "Facets and strong valid inequalities for integer programs," *Operations Research*, vol. 24, no. 2, pp. 367–372, 1976.

[105] E. Zemel, "Lifting the facets of zero–one polytopes," *Mathematical Programming*, vol. 15, no. 1, pp. 268–277, 1978.

[106] G. L. Nemhauser and L. E. Trotter, "Properties of vertex packings and independence system," *Mathematical Programming*, vol. 6, pp. 48–61, 1974.

[107] C. A. Meyer and C. A. Floudas, "Trilinear monomials with positive or negative domains: Facets of the convex and concave envelopes," in *Frontiers in Global Optimization* (C. Floudas and P. M. Pardalos, eds.), vol. 74 of *Nonconvex Optimization and Its Applications*, pp. 327–352, Boston, MA: Springer, 2004.

[108] M. W. Padberg, "On the facial structure of set packing polyhedra," *Mathematical Programming*, vol. 5, pp. 199–215, 1973.

[109] L. E. Trotter, "A class of facet producing graphs for vertex packing polyhedra," *Discrete Mathematics*, vol. 12, pp. 373–388, 1975.

[110] H. D. Sherali and J. C. Smith, "A polyhedral study of the generalized vertex packing problem," *Mathematical Programming*, vol. 107, no. 3, pp. 367–390, 2006.

[111] M. Colombi, R. Mansini, and M. Savelsbergh, "The generalized independent set problem: Polyhedral analysis and solution approaches," *European Journal of Operational Research*, vol. 260, no. 1, pp. 41–55, 2017.

[112] S. Hosseinian and S. Butenko, "Algorithms for the generalized independent set problem based on a quadratic optimization approach," *Optimization Letters*, vol. 13, no. 6, pp. 1211–1222, 2019.

[113] A. Dessmark, K. Jansen, and A. Lingas, “The maximum k-dependent and f-dependent set problem,” in *International Symposium on Algorithms and Computation*, pp. 88–97, Springer, 1993.

[114] A. Pyatkin, E. Lykhovyd, and S. Butenko, “The maximum number of induced open triangles in graphs of a given order,” *Optimization Letters*, vol. 13, no. 8, pp. 1927–1935, 2019.

[115] C. Berge, “Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind (zusammenfassung),” *Wissenschaftliche Zeitschrift, Martin Luther Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche*, vol. 10, pp. 114–115, 1961.

[116] M. Chudnovsky, N. Robertson, P. Seymour, and R.Thomas, “The strong perfect graph theorem,” *Ann. Math.*, vol. 164, pp. 51–229, 2006.

[117] M. Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour, and K. Vušković, “Recognizing berge graphs,” *Combinatorica*, vol. 25, no. 2, pp. 143–186, 2005.

[118] M. Chudnovsky, A. Scott, P. Seymour, and S. Spirkl, “Detecting an odd hole,” *Journal of the ACM (JACM)*, vol. 67, no. 1, pp. 1–12, 2020.

[119] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vušković, “Even-hole-free graphs part ii: Recognition algorithm,” *Journal of Graph Theory*, vol. 40, no. 4, pp. 238–266, 2002.

[120] H. Chang and H. Lu, “A faster algorithm to recognize even-hole-free graphs,” *Journal of Combinatorial Theory, Series B*, vol. 113, pp. 141–161, 2015.

[121] D. Bienstock, “On the complexity of testing for odd holes and induced odd paths,” *Discrete Mathematics*, vol. 90, no. 1, pp. 85–92, 1991.

[122] D. Bienstock, “Corrigendum to: On the complexity of testing for odd holes and induced odd paths,” *Discrete Mathematics*, vol. 102, no. 1, p. 109, 1992.

[123] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York: W.H. Freeman and Company, 1979.

[124] Y. Chen and J. Flum, "On parameterized path and chordless path problems," in *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity*, (Washington, DC), pp. 250–263, IEEE Computer Society, 2007.

[125] G. L. Nemhauser and L. E. Trotter, "Vertex packing: structural properties and algorithms," *Mathematical Programming*, vol. 8, pp. 232–248, 1975.

[126] H. D. Sherali and W. P. Adams, "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems," *SIAM Journal of Discrete Mathematics*, vol. 3, pp. 411–430, 1990.

[127] E. Boros and P. L. Hammer, "Pseudo-boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1-3, pp. 155–225, 2002.

APPENDIX A

THE UNIVARIATE LAGRANGIAN RELAXATION PROBLEM

Lemma 31. *Suppose S is a set of objects, each of which has a weight $w_i > 0,\ \forall i \in S$. For every $x \geq 0$, let $S^+ = \{i \in S \mid w_i \geq 2x\}$. Then, for every positive integer N, the optimal value of the following optimization problem is always given by sum of the N largest weights of the objects in S:*

$$\min_{x \geq 0} \sum_{i \in S^+} (w_i - 2x) + 2Nx. \tag{A.1}$$

Proof. Let S_N denote the set of N objects with the largest weights in S, and $\tilde{w}$ be the average weight of the objects in S_N, i.e., $\tilde{w} = \frac{1}{N}\sum_{i \in S_N} w_i$. First, we show that $N\tilde{w} \leq \sum_{i \in S^+}(w_i - 2x) + 2Nx$. Patently, the inequality holds if $\tilde{w} \leq 2x$. Hence, suppose $\tilde{w} > 2x$. Consider a partition of S_N as follows:

$$\begin{aligned}
&S_1 = \{i \in S_N \mid \tilde{w} \leq w_i\},\\
&S_2 = \{i \in S_N \mid 2x \leq w_i < \tilde{w}\},\\
&S_3 = \{i \in S_N \mid w_i < 2x\}.
\end{aligned} \tag{A.2}$$

Note that, $S_1 \cup S_2 = S_N \cap S^+$. By definition of $\tilde{w}$, we have

$$\begin{aligned}
\sum_{i \in S_1}(w_i - \tilde{w}) &= \sum_{i \in S_2}(\tilde{w} - w_i) + \sum_{i \in S_3}(\tilde{w} - w_i)\\
&= \sum_{i \in S_2}(\tilde{w} - w_i) + \sum_{i \in S_3}(\tilde{w} - 2x) + \sum_{i \in S_3}(2x - w_i)\\
&\geq \sum_{i \in S_2}(\tilde{w} - w_i) + \sum_{i \in S_3}(\tilde{w} - 2x),
\end{aligned} \tag{A.3}$$

as $w_i < 2x,\ \forall i \in S_3$. Therefore,

$$\begin{aligned}\sum_{i\in S^+}(w_i-2x) &\geq \sum_{i\in S_N\cap S^+}(w_i-2x) = \sum_{i\in S_1}(w_i-2x)+\sum_{i\in S_2}(w_i-2x)\\ &= \sum_{i\in S_1}(w_i-\tilde{w}+\tilde{w}-2x)+\sum_{i\in S_2}(w_i-\tilde{w}+\tilde{w}-2x)\\ &\geq \sum_{i\in S_1}(\tilde{w}-2x)+\sum_{i\in S_2}(\tilde{w}-2x)+\sum_{i\in S_3}(\tilde{w}-2x)\\ &= \sum_{i\in S_N}(\tilde{w}-2x) = N(\tilde{w}-2x),\end{aligned} \tag{A.4}$$

where the second inequality is due to (A.3). Since the choice of x was arbitrary, this result implies that

$$N\tilde{w} \leq \min_{x\geq 0} \sum_{i\in S^+}(w_i-2x)+2Nx. \tag{A.5}$$

Let x' be equal to one half of the smallest weight of an object in S_N. Then,

$$\sum_{i\in S^+}(w_i-2x')+2Nx' = \sum_{i\in S_N}(w_i-2x')+2Nx' = \sum_{i\in S_N} w_i - 2Nx' + 2Nx' = N\tilde{w}. \tag{A.6}$$

That is, by this choice of x, the univariate optimization problem reaches its lower bound. Hence, its optimal value is always given by $N\tilde{w} = \sum_{i\in S_N} w_i$. $\square$

APPENDIX B

SUPPLEMENTARY ALGORITHMS

Algorithm 12 Initialization step proposed in [49]

```
1: function INITIALIZE($G$)
2:     $U$ = an empty array of size $n$
3:     for $k = n$ to $1$ do                                        ▷ initial sorting starts here
4:         $R \leftarrow$ set of vertices with minimum degree in $G$
5:         if $|R|$ = 1 then
6:             $u \leftarrow$ the vertex in $R$
7:         else
8:             for every vertex $i \in R$ do
9:                 $\sigma(i) = \sum_{v \in N(i)} d_v$                  ▷ $d_v$: degree of vertex $v$
10:            end for                                       ▷ $N(i) = \{j \in V \mid \{i, j\} \in E\}$
11:            $u \leftarrow$ a vertex in $R$ with the minimum $\sigma$
12:        end if
13:        $U[k] \leftarrow u$
14:        $G \leftarrow G \backslash \{u\}$
15:    end for
16:    $L \leftarrow U$                                                ▷ initial coloring starts here
17:    for $k = 1$ to $\Delta(G)$ do
18:        color($L[k]$) $\leftarrow k$
19:    end for
20:    for $k = \Delta(G) + 1$ to $n$ do
21:        color($L[k]$) $\leftarrow \Delta(G) + 1$
22:    end for
23:    return $(U, L)$
24: end function
```

Algorithm 13 Vertex coloring method proposed in [49]

```
1: function SUBCOLOR^1$(G, U_v)$
2:     $L_v$ = an empty array of size $|U_v|$
3:     $K = 0$                                  ▷ $K$: the number of colors used
4:     $I_1 = \emptyset$
5:     for $i = 1$ to $|U_v|$ do
6:         $u \leftarrow U_v[i]$
7:         $k = 1$
8:         while $N(u) \cap I_k \neq \emptyset$ do
9:             $k \leftarrow k + 1$
10:        end while
11:        if $k > K$ then
12:            $K \leftarrow k$
13:            $I_k = \emptyset$
14:        end if
15:        $I_k \leftarrow I_k \cup \{u\}$
16:    end for
17:    $i = 1$
18:    for $k = 1$ to $K$ do
19:        for $j = 1$ to $|I_k|$ do
20:            $L_v[i] \leftarrow I_k[j]$
21:            color($L_v[i]$) $\leftarrow k$
22:            $i \leftarrow i + 1$
23:        end for
24:    end for
25:    return $L_v$
26: end function
```

Algorithm 14 Recoloring method proposed in [52]

```
1: function SUBCOLOR$^2(G, U_v, C, W, W^*)$
2:     $L_v$ = an empty array of size $|U_v|$
3:     $T = $ THRESHOLD$(G, U_v, C, W, W^*)$
4:     $F = \emptyset$                                   ▷ $F$: the set of forbidden colors
5:     $s = 1$ ; $I_1 = \emptyset$
6:     $Array \leftarrow U_v$
7:     while $Array \neq \emptyset$ do
8:         $I_s \leftarrow Array$
9:         while all vertices in $I_s$ have been selected do
10:            pick a vertex $v$ from $I_s$
11:            $result \leftarrow$ false
12:            if $s \geq T + 1$ then $result \leftarrow$ EXCLUDE$(v, T, \{I_1, \cdots, I_T\}, I_s, F)$
13:            if $result =$ false then $I_s \leftarrow I_s \backslash N(v)$
14:            $Array \leftarrow Array \backslash \{v\}$
15:        end while
16:        $s \leftarrow s + 1$ ; $I_s = \emptyset$
17:    end while
18:    $i = 1$
19:    for $k = 1$ to $s - 1$ do
20:        for $j = 1$ to $|I_k|$ do
21:            $L_v[i] \leftarrow I_k[j]$
22:            color($L_v[i]$) $\leftarrow k$
23:            $i \leftarrow i + 1$
24:        end for
25:    end for
26:    return $L_v$
27: end function
```

Algorithm 15 Vertex exclusion method proposed in [52]

```
function EXCLUDE($v, T, \{I_1, \cdots, I_T\}, I_s, F$)
    for $k_1 = 1$ to $T$ do
        if $k_1 \notin F$ and $|I_{k_1} \cap N(v)| = 1$ then
            $u \leftarrow I_{k_1} \cap N(v)$
            for $k_2 = k_1 + 1$ to $T$ do
                if $k_2 \notin F$ and $I_{k_2} \cap N(v) \cap N(u) = \emptyset$ then
                    $F \leftarrow F \cup \{k_1\} \cup \{k_2\}$
                    $I_s \leftarrow I_s \backslash \{v\}$
                    return true
                end if
            end for
            for $k_2 = 1$ to $k_1 - 1$ do
                if $k_2 \notin F$ and $I_{k_2} \cap N(v) \cap N(u) = \emptyset$ then
                    $F \leftarrow F \cup \{k_1\} \cup \{k_2\}$
                    $I_s \leftarrow I_s \backslash \{v\}$
                    return true
                end if
            end for
        end if
    end for
    return false
end function
```