OPTIMIZATION IN GEOMETRIC GRAPHS: COMPLEXITY AND APPROXIMATION

A Dissertation

by

SERA KAHRUMAN-ANDEROGLU

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

DOCTOR OF PHILOSOPHY

December 2009

Major Subject: Industrial Engineering

OPTIMIZATION IN GEOMETRIC GRAPHS: COMPLEXITY AND APPROXIMATION

A Dissertation

by

SERA KAHRUMAN-ANDEROGLU

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

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Sergiy I. Butenko
Committee Members,	Illya V. Hicks
	Lewis Ntaimo
	Vivek Sarin
Head of Department,	Brett A. Peters

December 2009

Major Subject: Industrial Engineering

ABSTRACT

Optimization in Geometric Graphs:
Complexity and Approximation. (December 2009)
Sera Kahruman-Anderoglu, B.S., Bogazici University
Chair of Advisory Committee: Dr. Sergiy I. Butenko

We consider several related problems arising in geometric graphs. In particular, we investigate the computational complexity and approximability properties of several optimization problems in unit ball graphs and develop algorithms to find exact and approximate solutions. In addition, we establish complexity-based theoretical justifications for several greedy heuristics.

Unit ball graphs, which are defined in the three dimensional Euclidian space, have several application areas such as computational geometry, facility location and, particularly, wireless communication networks. Efficient operation of wireless networks involves several decision problems that can be reduced to well known optimization problems in graph theory. For instance, the notion of a "virtual backbone" in a wireless network is strongly related to a minimum connected dominating set in its graph theoretic representation.

Motivated by the vastness of application areas, we study several problems including maximum independent set, minimum vertex coloring, minimum clique partition, max-cut and min-bisection. Although these problems have been widely studied in the context of unit disk graphs, which are the two dimensional version of unit ball graphs, there is no established result on the complexity and approximation status for some of them in unit ball graphs. Furthermore, unit ball graphs can provide a better representation of real networks since the nodes are deployed in the three di-

mensional space. We prove complexity results and propose solution procedures for several problems using geometrical properties of these graphs.

We outline a matching-based branch and bound solution procedure for the maximum k-clique problem in unit disk graphs and demonstrate its effectiveness through computational tests. We propose using minimum bottleneck connected dominating set problem in order to determine the optimal transmission range of a wireless network that will ensure a certain size of "virtual backbone". We prove that this problem is NP-hard in general graphs but solvable in polynomial time in unit disk and unit ball graphs.

We also demonstrate work on theoretical foundations for simple greedy heuristics. Particularly, similar to the notion of "best" approximation algorithms with respect to their approximation ratios, we prove that several simple greedy heuristics are "best" in the sense that it is NP-hard to recognize the gap between the greedy solution and the optimal solution. We show results for several well known problems such as maximum clique, maximum independent set, minimum vertex coloring and discuss extensions of these results to a more general class of problems.

In addition, we propose a "worst-out" heuristic based on edge contractions for the max-cut problem and provide analytical and experimental comparisons with a well known "best-in" approach and its modified versions.

ACKNOWLEDGMENTS

My appreciation goes to my advisor, Dr. Sergiy Butenko, for introducing me to the interesting problems in this dissertation and for his support and encouragement. I would also like to thank Dr. Illya Hicks, Dr. Lewis Ntaimo and Dr. Vivek Sarin for serving as members of my committee.

My gratitude goes to Judy Meeks, the Graduate Supervisor at the Department of Industrial & Systems Engineering. She was always friendly, helpful and supportive. From the Department, I also thank the Computer Systems staff member, Dennis Allen, for his willingness to help during the times that I had technical problems with computers.

I sincerely thank my parents, Fatma and Necdet Kahruman, and sisters for their love, faith and support during all these years. I also thank my parents-in-law, particularly my mother-in-law, Elif Anderoglu, for her support during the most crucial times of my study.

Finally, I would like to dedicate all of my work to my husband, Osman, and my daughter, Meryem. Without their love, faith and sacrifice, I would never have accomplished this work.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

In this dissertation, we study several related problems arising in geometric graphs. In particular, we investigate the computational complexity and approximability properties of several optimization problems in unit ball graphs and develop algorithms to find exact and approximate solutions. In addition, we establish a complexity-based theoretical justification for some simple construction heuristics.

A unit ball graph (UBG) is the intersection graph of a family of unit radius balls in the three-dimensional Euclidean space. Touching balls are assumed to be intersecting as well. This is called the *intersection model* for UBGs. A UBG can be viewed as the three-dimensional version of a unit disk graph (UDG) which is defined as the intersection graph of unit circles in the plane. An alternative way to describe UBGs is a *containment model*. Given a set of unit-radius balls in the three-dimensional Euclidian space, associate a vertex with each unit ball and form an edge between two vertices if and only if one of the corresponding unit balls contains the center of the other ball. Another alternative for the description of UBGs is a *proximity model*, in which, given the coordinates of a set of points in the three-dimensional Euclidian space, we form a vertex for each point and an edge between two vertices if and only if the Euclidian distance in between the corresponding points is less than or equal to a unit distance.

UDGs have several application areas such as computational geometry and facility location. Particularly, they have been widely used to model the topology of ad-hoc wireless communication networks. In this application each node represents a station.

The journal model is Mathematical Programming.

It is assumed that nodes have the same transmission radii and have omnidirectional antennas. Node locations are modeled as Euclidean points and the transmission areas are modeled as unit circles around these points. Several optimization problems on UDGs are solved in order to operate these networks effectively [12]. An example is the frequency assignment problem, which is to assign different frequencies to the nodes whose transmission ranges intersect. In UDG model, this problem is equivalent to the graph coloring problem. Another example is the routing problem on these networks, which involves the virtual backbone. The virtual backbone of a network is a subset D of nodes such that non-adjacent nodes can communicate with each other through the nodes in D. The size of a virtual backbone is desired to be as small as possible. This corresponds to the minimum connected dominating set problem in graph theory.

UDGs have been widely studied by many researchers since they provide a simple graph theoretic model, especially for wireless networks. Another motivation stated by Clark et al. [29] is that although many other intersection families were studied earlier in the literature yielding efficient algorithms, their efficiency was attributed to being a subclass of perfect graphs, for which efficient algorithms already exist for arbitrary graphs. However a UDG is not necessarily perfect.

Given a graph without its geometric representation, it is NP-hard to determine whether it can be represented as a unit disk graph or a unit ball graph [18].

Many NP-hard optimization problems remain NP-hard in UDGs. However, there exist approximation algorithms with constant approximation ratios. Moreover many of these problems allow for a polynomial time approximation scheme (PTAS). Marathe et al. [66] present several constant ratio heuristics for maximum independent set, maximum clique, minimum vertex cover, chromatic number and several types of domination problems. The geometric properties are used to determine the approximation ratios. Clark et al. [29] proved that the maximum clique problem, which is

one of the most famous NP-hard problems in general graphs, is solvable in polynomial time in UDGs. Jansen et al. [55] present PTAS for maximum bisection problem in planar graphs and unit disk graphs. Erlebach et al. [75] offer a PTAS for both maximum independent set and minimum vertex cover problems. Cheng et al. [26] propose a PTAS for minimum connected dominating set problem. There are several other papers addressing approximation algorithms [23, 36, 76, 38]. Majority of the above mentioned PTAS's use the so-called *shifting technique*. In this technique, a set of regularly spaced separators is used to decompose the problem into smaller, easier solvable subproblems. The solutions of the subproblems are merged to form a solution to the original problem. This is repeated for several placements of the separator set. The best solution over these placements is then selected as an approximation of the optimum.

UDG model is the simplest model used for sensor networks, thus there are oversimplifications and it is too optimistic as stated in [72]. The authors introduce several graph models used for sensor networks such as quasi-UDG, UBG in a doubling metric, and bounded independence graphs (BIG). The UBG, which is defined in the three-dimensional Euclidian space, serves as a better model in terms of overcoming some of the UDG-related simplifications.

Furthermore, there is increased interest in applications where ad hoc and sensor networks may be deployed in three-dimensional space, such as in an ocean, the atmosphere or in a building [34]. An example of ocean monitoring is presented in [5], where the nodes in the network have to be placed at different depths and thus create a three-dimensional network.

UDGs are a subclass of UBGs since given a unit disk representation of a graph we can easily find the unit ball representation by assigning the same value for the third coordinate. Thus, optimization problems that are proven to be NP-Hard in unit

disk graphs are also NP-hard in unit ball graphs.

The recognition of UBGs is also NP-hard [18]. It is even NP-hard to recognize a unit disk graph even if it does have a unit ball graph representation. The recognition of unit ball graphs restricted to the case where balls can only touch one another is also proven to be NP-hard [50].

There are very few papers addressing optimization problems in UBGs compared to the vast amount of work done for UDGs. Although many authors state that the algorithms they present for problems in UDGs can be extended to other intersection graphs and higher dimensions, the details, such as corresponding performance ratio and running time, are not provided.

Durocher et al. [34] present routing algorithms for mobile networks when they are represented as UBGs. Constant-ratio approximation algorithms for the maximum clique problem in UBGs is presented by Afshani and Chan in [2]. Zhang et al. [85] offer PTAS for the minimum connected dominating set problem in UBGs and state that the existing PTAS for the same problem in UDGs can not be directly extended to UBGs.

In this dissertation, we provide a survey of approximation algorithms for several problems in unit ball graphs, mostly by extending the techniques used for unit disk graphs. We also focus on problems such as the maximum k-clique and minimum bottleneck connected dominating set.

Given a graph G, a subgraph S of G is a *k-clique*, if the maximum pairwise distance of the vertices in S is at most k. The pairwise distance is the length of the shortest path (number of the edges on the shortest path) in between the pair of vertices. The *maximum k-clique problem* is to find a k-clique of maximum cardinality. Balasundaram et al. [13] proved that the k-clique problem is NP-hard for general graphs. We are interested in the computational complexity of the maximum

k-clique problem in unit disk graphs since the maximum clique problem is solvable in polynomial time in these graphs. Furthermore, clique relaxations such as k-clique may be more realistic compared to the clique when finding cohesive subgraphs. To our best knowledge, this problem has not been studied in unit disk graphs before.

Motivated by the wireless network applications, we propose the minimum bottleneck connected dominating set problem in order to find an optimal transmission range when there is a requirement on the size of the "virtual backbone" of the network. The transmission range of a node is directly related to its energy usage. Designing power-efficient networks is crucial. Thus, the choice of transmission range is an important decision problem. In the wireless networks community, there are several researchers who work on designing power-efficient networks. Some of these researchers address the optimal transmission range problem. They provide solution strategies with goals such as providing connectivity in the network. To our best knowledge, the goal of ensuring a certain size of "virtual backbone" has not been studied before in the wireless networks community. We focus on unit disk and unit ball graph models of wireless networks. We observe that this problem has not been studied in the graph theory literature as well. The bottleneck version of the dominating set problems studied focuses on vertex-weighted graphs where bottleneck cost is defined in terms of the vertex weights. Yen [84] introduces the bottleneck dominating set and bottleneck independent dominating set problems where the bottleneck is defined as the maximum weighted vertex. The author shows that this problem can be solved in $O(nlogn + m)$ time. On the other hand, the bottleneck independent dominating set problem is proven to be NP-hard on planar graphs. Kloks et al. [59] present linear time algorithms for minimum bottleneck dominating set and minimum bottleneck total dominating set problems. They also state that the minimum bottleneck connected dominating set problem can be solved in $O(mlogn)$ time. We study the edge-weighted

version of this problem and prove that it is NP-hard in general graphs.

We also study the theoretical justification for the choice of heuristics. In many cases, researchers and practitioners rely on variations of greedy heuristics that are very simple to understand and implement for solving NP-hard problems approximately. This simplicity and effectiveness of heuristic approaches earned them a considerable popularity in optimization community. However, there is also fair amount of skepticism towards such approaches due to a lack of theoretical foundations behind them. We propose complexity-based techniques that can be used to characterize "provably best" heuristics. When an optimization problem is said to be inapproximable within a factor of some constant $c - \epsilon$, it is easy to claim that if an approximation algorithm has a performance guarantee of c, it is the "best" possible. However, for many problems such as maximum clique, the inapproximability result is stated in terms of the problem size. Thus, we cannot claim a heuristic to be the "best" with respect to its approximation ratio. Thus, we need an alternative way to justify the choice of a certain heuristic. This problem has not been studied in the literature before.

Another research topic in this dissertation is the comparison of several greedy heuristics for the max-cut problem in general graphs by computational experiments. In the next section, we give a list of our research objectives and then we summarize our contributions.

I.1. Objectives

Motivated by the applications in wireless networks, we investigate several optimization problems in unit disk and unit ball graphs. The overall goal is to develop and improve solution procedures for these problems. This research comprises a series of linked objectives. These are:

1. Survey optimization problems in unit disk graphs and work on extensions of the existing algorithms to unit ball graphs;

2. Identify open problems in unit disk and unit ball graphs regarding computational complexity and approximation status and propose solutions strategies;

3. Develop and implement an exact solution procedure for the maximum *k-clique problem* in unit disk graphs;

4. Analyze and develop a centralized approximation algorithm for the *minimum k-bottleneck connected dominating set problem* in unit disk and unit ball graphs;

5. Establish complexity-based techniques for analysis of heuristics to provide a theoretical justification for the choice of construction heuristics;

6. Perform experimental comparison of several heuristics for the max-cut problem in general graphs.

I.2. Contributions

We analyze the complexity and approximation status of several optimization problems in UBGs. Furthermore, we identify several interesting open problems related to complexity and approximability of some optimization problems. We study the maximum k-clique problem in unit disk graphs and propose an efficient exact solution procedure. This problem has not been studied in unit disk graphs before. We also propose using bottleneck connected dominating set problem in order to determine an optimal transmission range for the nodes of a wireless network. This is an important decision problem for which, to our best knowledge, there has not been any research done with the same design goal. The transmission range of a wireless node is directly related to the energy usage. Thus, it is important to minimize energy usage while efficiently

operating the network. Our goal is to find the minimum transmission range such that we can have a "virtual backbone" of a certain size. We analyze the complexity of this problem in general graphs and unit disk graphs and also provide a 3-approximation algorithm for graphs whose edge weights satisfy triangle inequality. We prove that this problem is NP-hard in general graphs and polynomial-time solvable for unit disk graphs for any given constant "virtual backbone" size.

Heuristics are widely used to tackle large scale NP-hard problems. They are usually simple to understand and implement. However, they are criticized for lacking theoretical justification. For approximation algorithms, the established approximation ratios and results related to inapproximability give insight about whether an approximation algorithm is the "best" one. We establish theoretical justifications for the choice of simple heuristics for several optimization problems that are hard to approximate within *any* constant ratio. We use complexity-based techniques to prove that for many optimization problems it is NP-hard to find a better solution than a simple greedy heuristic in polynomial time. We believe that this is an important contribution since, to our best knowledge, there is no research published on theoretical justifications of greedy heuristics in a broad sense.

We study the max-cut problem in general graphs and experimentally compare the performance of several greedy heuristics. The existing methods mostly rely on choosing the best candidate at each iteration. We propose a "worst-out" approach and show that it does not perform better than a well-known "best-in" heuristic experimentally. We also test the performance of different variations and combinations of these heuristics. We observe that, although some heuristics have the same approximation ratio in theory, one of them may provide better solutions in practice.

Overall we contribute to the study of geometric graphs by establishing complexity and approximability results as well as introducing new optimization problems which

can be very useful in wireless network applications. We also contribute to optimization methodology by providing complexity-based techniques for analysis of heuristics.

I.3. Organization

Chapter II reviews the basic graph definitions used throughout this dissertation. In Chapter III, we analyze the complexity and approximation status of several optimization problems in unit ball graphs. We also present the developed approximation algorithms. The first part of Chapter III presents important geometric properties of unit disk and unit ball graphs. Some of these properties are extended from the two-dimensional case. These properties are used in the analysis of approximation algorithms. Each section of Chapter III focuses on a different optimization problem. We present the literature review for these problems in unit ball graphs accompanied by the extensions from unit disk graphs. The considered problems include domination problems, max-cut, max bisection, min bisection, maximum independent set, minimum vertex coloring and minimum clique partition. We present complexity results as well as approximation algorithms. We highlight several interesting open problems.

Chapter IV focuses on the maximum k-clique problem. First, we present a review of related literature. We claim the NP-hardness of the maximum k-clique problem in unit disk graphs as a conjecture. We present results on computational complexity of several related problems which can be helpful in the computational complexity analysis of the maximum k-clique problem in unit disk graphs. We outline a sequential greedy heuristic for general graphs. Using the fact that the maximum clique problem is polynomial-time solvable in unit disk graphs, we propose a matching-based branch and bound method for the exact solution of the maximum k-clique in unit disk graphs. The details of this exact solution procedure, as well as implementation details are

discussed in this chapter. Finally, Chapter IV presents computational results on randomly generated unit disk graphs. Since a k- clique is a clique on the k^{th} power of the input graph, we use the 1-plex formulation of Balasundaram [10] for comparison purposes.

Chapter V starts with the motivation for and the definition of the minimum bottleneck connected dominating set problem. A literature review on transmission range optimization in wireless networks and bottleneck dominating set problems in graph theory is presented. The next section of Chapter V presents an analysis of the computational complexity of the problem. We prove that it is NP-hard in general graphs, as well as in graphs whose edge weights satisfy triangle inequality. We show that it is not possible to approximate our problem within a factor of $2 - \epsilon$ for any $\epsilon > 0$ when the input graph edge weights satisfy triangle inequality. We present a 3-approximation algorithm for graphs with triangle inequality. Finally, we show that this problem is polynomial-time solvable in unit disk graphs.

Chapter VI presents the results on theoretical justification of simple greedy heuristics. Motivated by the result we get on k-club heuristic, we investigate whether the simple greedy heuristics for several problems can be proved to be the "best" based on the complexity of the gap recognition between the greedy solution and the optimal solution values. We prove that we can not have a polynomial time algorithm provably always better than a simple greedy heuristic for several problems such as maximum clique, maximum independent set, minimum vertex color, minimum clique partition and maximum k-plex. Finally, we discuss if the same procedure can be applied to a broader class of optimization problems. We consider the node deletion problems which aim to find a maximum cardinality subgraph that satisfies a given hereditary, additive (or co-additive), interesting and nontrivial graph property π. This class includes several well known optimization problems such as maximum planar sub-

graph, maximum outerplanar subgraph, maximum clique, etc. We conjecture that for all problems in this category, it is impossible to have a polynomial time algorithm which always guarantees a better solution than a simple maximal by inclusion greedy heuristic unless $P = NP$. We present some arguments that can be used in the proof.

In Chapter VII, we present an edge contraction heuristic for the max-cut problem. We give a literature review on the existing approximation algorithms and heuristics for the max-cut problem. We analyze the approximation ratio of the new heuristic. Next, we present a comparison of several heuristics and their variations based on computational experiments.

Finally, Chapter VIII is reserved for conclusions and ideas for future work.

CHAPTER II

PRELIMINARIES

A graph is a pair $G = (V, E)$ of sets such that $E \subseteq V \times V$; thus the elements of E are 2-element subsets of V. The elements of V are the *vertices* (or *nodes*) of G, the elements of E are its *edges*. A graph with a vertex set V is said to be a graph *on* V. The vertex set of a graph G is denoted by $V(G)$, and its edge set by $E(G)$. The number of vertices of a graph G is its *order*, denoted as $|G|$.

Two vertices x, y of G are *adjacent*, *incident*, or *neighbors*, if xy is an edge of G. They are called the *endpoints* of the edge xy. The *degree* of a vertex is the number of edges incident with that vertex. The maximum degree in a graph G is denoted by $\Delta(G)$. G is called *complete* if all of its vertices are pairwise adjacent. A complete graph on n vertices is denoted by K^n.

A *walk* in a graph is a sequence of vertices such that from each of its vertices there is an edge to the next vertex in the sequence. A *path* is a walk with no repeated vertices. A *cycle* is a closed walk with no other repeated vertices than the starting and ending vertices. The *distance* $d_G(x, y)$ in G of two vertices x, y is the length of a shortest x-y path in G. The greatest distance between any two vertices in G is the *diameter* of G.

A graph is *connected* if every pair of vertices can be joined by a path. It is said to be a *k-connected* graph if there does not exist a set of $k-1$ vertices whose removal disconnects the graph. A maximal connected subgraph of G is called a *component* of G.

A graph $G = (V_1, E_1)$ is a *subgraph* of the graph $G = (V, E)$ if $V_1 \subseteq V$ and $E_1 \subseteq E$. For a subset $V' \subseteq V$, $G[V']$ denotes the graph *induced* by V', which is given

by $G[V'] = (V'; E \cap (V \times V'))$. A subgraph $G' \subseteq G$ is called a spanning subgraph of G if $V(G') = V(G)$.

A graph that does not contain any cycles is called a *forest*. It is also called *acyclic*. A connected forest is called a *tree*. The *leaves* of a tree are the vertices of degree one. A *minimum spanning tree* in a graph G is a tree that spans all the vertices of G and has the minimum total edge weight.

A *bipartite* graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V. It is called a *complete bipartite* graph, $K_{|U|,|V|}$, if there exist an edge joining any vertex in U to any vertex in V. A bipartite graph can be equivalently described as a graph that does not contain any odd length cycles. Similarly a *k-partite* graph is a graph whose vertices can be divided into k disjoint sets, $V_1, \ldots, V_k$, such that every edge connects a vertex in V_i to one in V_j for some j such that $i \neq j$.

Contraction of an edge e means deleting that edge and identifying the ends of e into one node. A *minor*of a graph G is a graph obtained from G by first deleting some vertices and edges, and then contracting some further edges. A graph is called *planar* if it can be embedded in the plane such that no two edges or vertices cross one another. It is well known that a graph is planar if and only if it does not contain K^5 or $K_{3,3}$ as a minor. A graph is called *outerplanar* if it has an embedding in the plane such that the vertices lie on a fixed circle and the edges lie inside the disk of the circle and don't intersect. For $k > 1$ a planar embedding is *k-outerplanar* if removing the vertices on the outer face results in a $(k-1)$-*outerplanar* embedding. A graph is *k-outerplanar* if it has a k-outerplanar embedding.

A *tree decomposition* is a mapping of a graph into a tree. The *treewidth* measures the number of graph vertices mapped onto any tree node in an optimal tree decomposition.

The k^{th} *power* of a graph G is a graph with the same set of vertices as G and an edge between two vertices if and only if there is a path of length at most k between them. The *complement* of a graph G is a graph $\overline{G}$ on the same vertices such that two vertices of $\overline{G}$ are adjacent if and only if they are not adjacent in G.

A *comparability graph* is an undirected graph that connects pairs of elements that are related to each other in a *partial order*. Comparability graphs have also been called *transitively orientable graphs*, *partially orderable graphs*, and *containment graphs*. A transitive orientation consists of an assignment of a direction to each edge of the graph such that the resulting directed graph satisfies a *transitive law*: whenever there exist directed edges (x, y) and (y, z), there must exist an edge (x, z). A *co-comparability graph* is a graph whose complement is a comparability graph.

An *isomorphism* of graphs G and H is a bijection $f : V(G) \to V(H)$ between the vertex sets of G and H such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H. A class of graphs that is closed under isomorphism is called a *graph property*. For instance, "containing a triangle" is a graph property. Thus, if G contains three pairwise adjacent vertices then so does every graph isomorphic to G.

Given an undirected graph G, a *dominating set* D is a subset of vertices of G such that every vertex of G is either in D or has a neighbor in D. A *proper coloring* of G is one in which every vertex is colored such that no two vertices of the same color are adjacent. G is said to be *k-colorable* if it admits a proper coloring with k colors. The graph coloring problem is to find a proper coloring with the least number of colors which is called the *chromatic number* of the graph.

For a graph $G = (V, E)$, a *cut* is a partition of the set if vertices V into two subsets V_1 and V_2. Any edge $(u, v) \in E$ with $u \in V_1$ and $v \in V_2$ is said to be crossing the cut and is a *cut edge*. The *size of a cut* is the total number of edges crossing the

cut. In edge weighted graphs, the size of the cut is defined to be sum of weights of the edges crossing the cut. The *max-cut problem* is to find a cut with maximum size. The variation in which each partition set is required to have the same cardinality is called the *max-bisection problem*. A partition of vertices of a graph into two equal-cardinality sets is called a *bisection*. The *min-bisection problem* is to find a bisection with minimum number of edges with endpoints in different partition sets.

An *independent set* of a graph is a subset of mutually non-adjacent vertices. The *maximum independent set problem* is to find an independent set of maximum cardinality. A *vertex cover* for an undirected graph G is a subset S of its vertices such that each edge has at least one endpoint in S. Given an undirected graph G, a *clique* is a subset of pairwise-adjacent vertices in G. The *maximum clique problem* is to find a clique of largest cardinality in G. The size of a maximum clique is called the clique number of G and is denoted by $\omega(G)$. The *minimum clique partition problem* is to partition a given graph G into a minimum number of cliques.

A *perfect graph* is a graph in which the chromatic number of every induced subgraph equals the clique number of that subgraph.

Given a graph $G = (V, E)$, a *matching* M in G is a set of pairwise non-adjacent edges (no two edges share a common vertex). A vertex is *matched* if it is incident to an edge in the matching. Otherwise the vertex is *unmatched*. A *maximal matching* is a matching M of a graph G with the property that if any edge not in M is added to M, it is no longer a matching. Given a *matching* M, an *alternating path* is a path in which the edges belong alternatively to the matching and not to the matching. An *augmenting path* is an alternating path that starts from and ends on unmatched vertices. One can prove that a matching is maximum if and only if it does not have any augmenting path.

Breadth-first search (BFS) and depth-first search (DFS) are frequently used

traversal techniques for graphs. DFS progresses by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it hasn't finished exploring. BFS visits the vertices of G uniformly across the breadth of the frontier of its search, visiting all vertices distance d from the source vertex s before looking for vertices at distance $d + 1$.

An approximation algorithm $\mathcal{A}$ for a minimization problem Π has an approximation ratio (or performance guarantee) ρ if $\mathcal{A}_x \leq \rho \times opt(x)$ for every instance x of Π with an optimal value $opt(x)$, where $\mathcal{A}_x$ denotes the output of algorithm $\mathcal{A}$ for instance x. If the problem is a maximization problem, then we have $\mathcal{A}_x \geq opt(x)/\rho$ for every instance x. $\mathcal{A}$ is a polynomial-time approximation scheme (PTAS) for Π if and only if for every instance x of Π and for any fixed $\epsilon > 0$, $\mathcal{A}$ runs in time polynomial in $|x|$ and delivers a solution that is within a factor ϵ of being optimal. For a PTAS, the running time can be exponential in $1/\epsilon$. Π is called APX-hard if there exists no PTAS for Π.

CHAPTER III

APPROXIMATION AND COMPLEXITY IN UBGs

Several optimization problems have been widely studied in unit disk graphs. Although unit ball graphs, defined in the three-dimensional Euclidian space, serve as a better model in real-life applications, they have not been studied in detail in the literature. The key in the analysis of algorithms for unit disk graphs is the establishment of some geometric properties. Thus, this chapter starts with establishment of some properties for both unit disk and unit ball graphs. Next, we present a survey for optimization problems studied in UDGs and outline the extensions for UBGs. We also present the approximation ratios and running times of these algorithms. The optimization problems studied include domination problems, minimum vertex cover, maximum independent set, maximum clique, coloring, max-cut, max bisection, min bisection and minimum clique partition. A section is reserved for each one of these problems. We highlight the open problems related to each one of these optimization problems. Finally, we give a summary of open problems at the end of this chapter.

It is easy to see that any induced subgraph of a unit ball graph is also a unit ball graph. Furthermore unit disk graphs are a subclass of unit ball graphs since given a unit disk representation of a graph we can easily find the unit ball representation by assigning the same value for the third coordinate. Thus, optimization problems that are proven to be NP-hard in unit disk graphs are also NP-hard in unit ball graphs.

Given a graph without its geometric representation, it is NP-hard to determine whether it can be represented as a unit disk graph or a unit ball graph [18]. Authors prove complexity by a reduction from the 3-SAT problem. Furthermore, their proof also implies that it is NP-hard to recognize a unit disk graph even if it does have a

unit ball graph representation. The recognition of unit ball graphs restricted to the case where balls can only touch one another is also proven to be NP-hard in [50].

III.1. Properties of Unit Disk and Unit Ball Graphs

Lemma 1. *Let G be a unit ball graph. For any vertex v in G, the neighborhood of the vertex v contains at most 12 independent vertices.*

Proof. This can be proved by a geometric concept called "kissing number". Kissing number is the maximum number of non-intersecting spheres that can touch a central sphere in N-dimensional space simultaneously. Kissing number for 3-dimensional space is 12. Thus, 12 serves as an upper bound on the number of independent vertices in the neighborhood of any vertex in a unit ball graph. The best upper bound would correspond to a slightly different definition of kissing number, particularly the maximum number of non-touching spheres that can touch a unit sphere simultaneously. Although the drawings for kissing number in 3-dimensional space have gaps, it is not clear whether we can have all non-touching spheres. Recently Durocher et al. [34] claim that we can fit at most 11 non-touching spheres and show this by a drawing of 12 points on the surface of a sphere and showing that some of these points are adjacent. However, this argument does not seem to be sufficient for a proof. □

Lemma 2. *Let v be the vertex with the smallest X-coordinate in a unit ball graph G. Then the size of the maximum independent set in $G(N(v))$ is at most 8.*

Proof. The main idea is similar to the case when G is a unit disk graph, which is presented by Marathe et al. [66]. The question is to find the maximum number of non-intersecting unit balls, each of which intersects with the unit ball centered at v. Let α denote this number and w_x denote the X-coordinate of vertex w. It is easy to see that $5 < \alpha < 12$. Restricting the problem to the set $N(v) \cap \{w \in V(G) : w_x = v_x\}$,

our problem reduces to finding the maximum number of non-intersecting unit disks, each of which intersects with the unit disk centered at v. This number is proven to be 5 in [66]. Hence, it corresponds to a lower bound for α.

Now suppose $\alpha = 9$. We can have at most 5 unit balls whose centers are located at v_x. Then the remaining 4 unit balls can be placed on one side of the unit ball centered at v_x, so that they do not intersect with the previous 5. This is a contradiction with kissing number being 12. For any unit ball w, we would be able place 5 unit balls whose centers are located at the same X-coordinate as of w and 4 on one side and 4 on the other side. Then the kissing number would be 13. Thus $\alpha < 9$, which yields an upper bound of $\alpha \leq 8$. □

The following lemma describes a property stated by Clark et al. [29] to prove polynomial time solution of the maximum clique problem in unit disk graphs.

Lemma 3. *Let G be a unit disk graph and u and v be any two adjacent vertices in G. Let d_{uv} be the distance between them. For a vertex w, let $N_d[w]$ denote the set of all vertices that are at distance at most d from w, including w itself. Denote by $S = \{N_{d_{uv}}[u] \bigcap N_{d_{uv}}[v]\}$. Then $\overline{G}[S]$ is a bipartite graph.*

We refer the reader to Corollary 3.2 of [29] for the details of the proof.

Lemma 4. *If G is a unit disk graph, then it does not contain induced subgraphs isomorphic to $K_{2,3}$ and $K_{1,6}$.*

Proof. Let G be a unit disk graph. Assume that G contains an induced subgraph isomorphic to $K_{2,3}$. Let the two non-adjacent vertices located at points u and v and the three nonadjacent vertices located at t_1, t_2 and t_3 correspond to the vertices of this subgraph. Figure 1 depicts the intersection of two unit disks centered at points u and v, respectively. The three common neighbors have to be located in this intersection.

Points A and B represent the intersection of the boundaries. The triangles uAv and uBv are symmetric and since there are three neighbors that are non-adjacent, two of them have to be located in one of these triangles and also in the intersection area. Let t_1 and t_2 be the points representing these vertices. Since $dist(u, v) > 2$ it is easy to observe that the angle $a < 60^o$. Then, by the cosine rule $dist(t_1, t_2) < 2$ which is a contradiction since vertices corresponding to these points are non-adjacent. Thus, G cannot contain an induced subgraph isomorphic to $K_{2,3}$.

Marathe et al. [66] prove that a vertex can have at most 5 independent neighbors if the graph is a unit disk graph. Thus, G cannot contain $K_{1,6}$ as an induced subgraph either. □

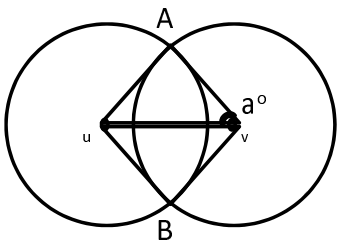

Fig. 1 Intersection of 2 unit disks centered at points v and u.

Lemma 5. *If G is a unit ball graph, then it does not contain induced subgraphs isomorphic to $K_{1,13}$, $K_{2,6}$ and $K_{3,4}$.*

Proof. Let G be a unit ball graph. It is easy to see that G cannot contain $K_{1,13}$ as an induced subgraph by Lemma 1.

Let u and v be two non-adjacent vertices in G. Consider the intersection of the unit balls representing these vertices. Afshani and Chan [2] show that the intersection area of unit balls of two adjacent vertices can be covered by a special shape called "rounded diamond", where vertices whose unit ball centers are located in the same

rounded diamond form a clique. There are on average 5.106 rounded diamonds covering this region. Since the intersection area of non-adjacent vertices will be smaller than that of adjacent vertices, 5.106 serves as an upper bound and thus we can say that we can fit at most 5 independent vertices here at the intersection. This proves that G cannot contain $K_{2,6}$ as an induced subgraph. Now observe that the intersection area of the unit balls of three non-adjacent vertices will be less than one half of the intersection of any two of them. Thus it is easy to see that we cannot fit 4 independent vertices in this area. This proves that G cannot contain $K_{3,4}$ as an induced subgraph. □

Lemma 6. *Given a unit disk graph, the k^{th} power of the graph is not necessarily a unit disk graph for any $k \geq 2$ if we are not allowed to change the coordinates of the points in the proximity model of the graph.*

Proof. We prove this lemma by a counterexample. Suppose the statement in the lemma is not true. Now suppose that a set of points is given together with their coordinates in Euclidian plane as in Figure 2. By specifying a unit distance of "1", we obtain the unit disk graph G in Figure 3. Next, we take the k^{th} power of G. We observe that, with the given coordinates, the vertices corresponding to the points a, b and c are all pairwise adjacent in G^k for any $k \geq 2$. The vertex that corresponds to the point d is always an isolated vertex. If G^k is a UDG, than it must have a unit distance. Since vertices corresponding to the points a and c are adjacent in G^k, the unit distance has to be at least 2. This is a contradiction with the definition of a UDG since, although the pairwise distance between c and d is always less than 2, the corresponding vertices can never be adjacent in G^k for any integer $k \geq 2$. □

The condition in Lemma 6 is a realistic assumption in terms of real-life applications. For instance, when we consider a wireless network, the locations of the wireless

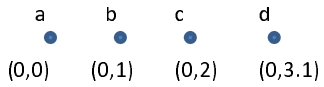

Fig. 2 The coordinates of four points in the plane.

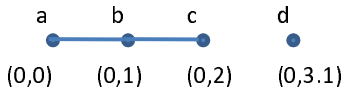

Fig. 3 The unit disk graph G with unit distance $= 1$.

nodes will be fixed for many applications. An interesting question is whether there exists a unit disk representation for the k^{th} power of a unit disk graph. Although the coordinates of the network may be fixed for a given application, finding another set of coordinates that will represent the power of the graph as a unit disk graph may be helpful in designing and analyzing algorithms to solve optimization problems on these graphs. For instance, for the example presented in Figure 2, Figure 3 and Figure 4, we can easily find a UDG representation for any power of the input graph. Consider moving the point d to $(0, 6)$. Then G^k is a UDG for any integer $k \geq 2$ with a unit distance of 2. The following lemma shows that this is not always true.

Lemma 7. *The k^{th} power of a unit disk graph does not necessarily have a unit disk graph representation for any $k > 1$.*

Proof. Assume that the statement is not true. For $k = 2$, consider the unit disk graph G in Figure 5. The points represent the centers of unit disks corresponding to the vertices of G. The solid lines are edges whose length is exactly the unit distance 1. The dashed lines are also edges but their length is less than 1. The figure is drawn to

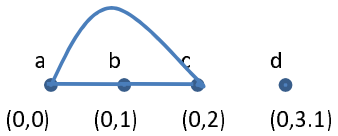

Fig. 4 k^{th} power of G for any integer $k > 0$.

scale. It is easy to see that these are the only edges in G since the distance between any two points that are not connected by either the dashed lines or the solid lines is greater than the unit distance. Now consider the 2^{nd} power of this graph and observe that the subgraph induced by vertices a, b, c, d and e forms $K_{2,3}$. Based on Lemma 4, G^2 is not a unit disk graph.

For an arbitrary k, consider the graph in Figure 6. Similar to the graph in Figure 5, all solid lines represent edges of length exactly equal to the unit distance of 1 and the dashed lines correspond to the edges whose length is less than the unit distance. It is easy to see that in G^k, the subgraph induced by vertices a, b, c, d and e forms a $K_{2,3}$. Thus G^k is not a UDG. □

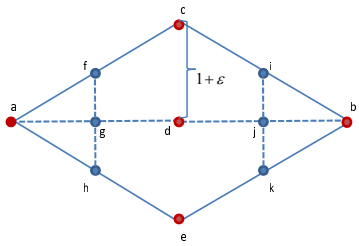

Fig. 5 Depicting a unit disk graph whose 2^{nd} power is not a unit disk graph.

For the rest of this chapter, we assume that the unit distance is 2 for the proximity model. That means that in the intersection model the unit disks have a radius of 1

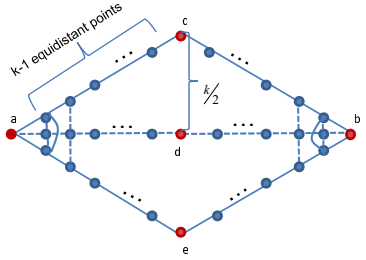

Fig. 6 Depicting a unit disk graph whose k^{th} power is not a unit disk graph for $k > 2$.

and in the containment model they have a radius of 2.

(k, l)-slab: A unit ball graph is a $(k, l)-$slab if the y-coordinates of all the centers of unit balls are contained in the interval $[0, k)$ and z-coordinates are contained in the interval $[0, l)$.

Theorem 1. *Let G be a unit ball graph which is a $(k, l)-$slab. G is a co-comparability graph if $k^2 + l^2 < 3$.*

Proof. The proof is similar to the case when G is a unit disk graph [67]. Let G be a unit ball graph with the above property. Consider the complement of G which is denoted by $\overline{G}$. We can assign directions to the edges of $\overline{G}$ such that the resulting graph satisfies the transitive law which will prove that $\overline{G}$ is a comparability graph. Let v_1, v_2 and v_3 be three vertices (center points of unit balls) in G. Transitive law requires that if there is an edge from v_1 to v_2 and an edge from v_2 to v_3, then there exists an edge from v_1 to v_3. Let x_i, y_i and z_i denote the x, y and z coordinates of point v_i for each i. Without loss of generality, we can assume that $x_1 < x_2 < x_3$. Let (v_1, v_2) be an edge of $\overline{G}$. Thus $(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 > 4$. We direct the edge from v_1 to v_2. Since $(y_1 - y_2)^2 + (z_1 - z_2)^2 < 3$, we have $x_1 + 1 < x_2$. Similarly, for the edge (v_2, v_3) we have $x_2 + 1 < x_3$. Hence $x_1 + 2 < x_3$, which implies that

(v_1, v_3) is an edge of $\overline{G}$. □

Now a survey and extensions of existing algorithms for UBGs are in order.

III.2. Domination Problems

Given an undirected graph G, a *dominating set* D is a subset of vertices of G such that every vertex of G is either in D or has a neighbor in D. Minimum dominating set problem is to find a dominating set of minimum size. With additional restrictions on set D, we have the following variations: Minimum independent dominating set when D has to be an independent set, minimum connected dominating set when $G[D]$ is connected and minimum total dominating set when the vertices in D are also required to have a neighbor in D.

Marathe et al. [66] investigate efficient heuristics for minimum dominating set, minimum independent dominating set, minimum connected dominating set, and minimum total dominating set in unit disk graphs. Constant performance ratio of 5 is achieved for the first two problems while the latter two have 10-approximations. Since a maximal independent set is also a dominating set, the relation between the size of a maximal independent set and that of a dominating set is the key property used in the performance analysis of these algorithms. Thus, all of these algorithms in [66] can be applied on UBGs as well with different performance ratios.

Theorem 2 ([66])**.** *Let G be a unit disk graph. Let D^* be a minimum dominating set for G and let D be any maximal independent set for G. Then $|D| \leq 5|D^*|$.*

It is easy to see that any vertex in D^* can have at most 5 independent neighbors in a unit disk graph. Thus, the size of any maximal independent set is bounded by $5|D^*|$. Theorem 2 can be easily extended to unit ball graphs as well by Lemma 1.

Corollary 1. *Let G be a unit ball graph. Denote by D^* and D the minimum dominating set and any maximal independent set of G, respectively. Then $|D| \leq 12|D^*|$.*

Hence, the algorithms presented by Marathe et al. [66] have the following performance ratios when the input graph is a UBG: minimum dominating set and minimum independent dominating set problems have a performance ratio of 12 while minimum connected dominating set and minimum total dominating set problems have 24−approximations.

The performance guarantee for the minimum connected dominating set approximation in unit disk graphs is improved to 7.6 by the following observation:

Theorem 3 ([83])**.** *For any unit disk graph G, the size of a maximal independent set is at most $3.8|mcds(G)| + 1.2$, where $mcds(G)$ is a minimum connected dominating set of G.*

This theorem is established by using the following properties:

Lemma 8 ([83])**.** *The neighbor area of two adjacent vertices contains at most 8 independent vertices in a unit disk graph.*

Lemma 9 ([83])**.** *For any unit disk graph, there exists a minimum weight spanning tree such that every vertex has degree at most 5.*

Performance guarantee for the minimum connected dominating set approximation of Marathe et al. [66] in unit disk graphs is further improved to 6.91 by Funke et al. [38] with the following observation:

Theorem 4 ([38])**.** *The size of any independent set in a unit disk graph G is at most $3.453|mcds(G)| + 8.291$, where $mcds(G)$ denotes a minimum connected dominating set in G.*

Funke et al. [38] achieve this relation by an analysis of the area covered by the connected dominating set. More specifically, the disks of radius 3 and with the same centers as the centers of the unit disks forming a dominating set, cover all the unit disks in the graph. Furthermore, since it is a connected dominating set, there are overlaps of these larger disks. Next, they determine how many non-overlapping unit disks can be placed in this area by using the well-known result of Fejes Tóth, which proves that the densest packing of unit disks in the plane is achieved by a hexagonal lattice.

Extending the technique used in [38] to unit ball graphs, we observe a tighter bound on the cardinality of a maximal independent set in terms of the cardinality of a minimum connected dominating set. In this case, we deal with the volume covered by the balls with radius 3. Clearly, this volume contains all the unit balls.

Theorem 5. *Let G be a unit ball graph. The volume covered by the union of unit balls in G is at most $54.455|mcds(G)| + 58.643$, where $mcds(G)$ is the minimum connected dominating set in G.*

Proof. Given a unit ball graph G, let S be the set of balls with radius 3 centered at the centers of the unit balls from the minimum connecting dominating set of G. Since the dominating set is connected, there exists an ordering $s_1, s_2, \ldots, s_{|S|}$ of the balls in S such that the center of s_i is at distance at most 2 from the center of some s_j with $j < i$. To bound the volume covered by S, we follow the order of these balls. At iteration $i > 1$, the volume increases by less than the volume of a single ball since there are overlaps. Since the volume of overlap of two balls of radius r and distance d between their centers is given by $\frac{1}{12}\pi(4r + d)(2r - d)^2$, the volume increase V^+ at each iteration i has the following upper bound:

$$V^+ \leq \frac{4}{3}\pi r^3 - \frac{1}{12}\pi(4r + d)(2r - d)^2,$$

where d is the distance between the centers of s_i and the closest to s_i ball among the balls $s_1, \ldots, s_{i-1}$. Thus, for $(r, d) = (3, 2)$ this equation yields an upper bound: $V^+ \leq 54.455$. Therefore, the total volume covered by S is at most $54.455(|mcds(G)|-1) + 113.098 = 54.455|mcds(G)| + 58.643$ □

It is easy to see that, by using the volume of a single unit ball we obtain the following relation between a maximal independent set IS and the minimum connected dominating set $mcds$ in a unit ball graph: $|IS| \leq 13|mcds| + 27$. This bound is worse than the bound in Corollary 1. However, it is obvious that a close packing of the unit balls will have gaps in between. Thus, we are interested in the actual volume covered by a unit ball in the densest packing. This had been an open problem for a very long time. Kepler proposed his famous conjecture in 1611 stating that close packing (either cubic or hexagonal close packing, both of which have maximum densities of $\frac{\pi}{\sqrt{18}} \simeq 74.048\%$) is the densest possible sphere packing in 3 dimensional Euclidian space. His conjecture was proved by Thomas C. Hales in 2000, which took 282 pages and extensive computer calculations [46]. It is reported that this face-centered cubic packing is produced by placing a ball inside each rhombic dodecahedron in the tiling. A rhombic dodecahedron (Figure 7) is a convex polyhedron with 12 rhombic faces. It has 14 vertices and 24 equal-length edges.

Theorem 6. *The size of any independent set in a unit ball graph G is at most* $11.23|mcds(G)| + 23.33$.

Proof. Given a unit ball graph G, by Theorem 5 the volume of the union of unit balls in G, (V_G), is at most $54.455|mcds(G)| + 58.643$. The densest packing of unit balls is achieved by placing a ball inside each rhombic dodecahedron. Therefore, the number of rhombic dodecahedrons that can fit in V_G is an upper bound on the size of any independent set in G. It is important to note that, although all the

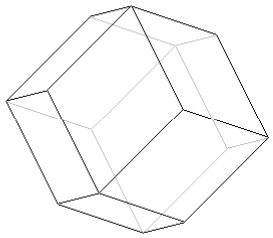

Fig. 7 A rhombic dodecahedron.

unit balls forming the independent set lie in V_G, the rhombic dodecahedra containing them may not be completely contained in V_G. Thus, we add a correction factor to the volume of each rhombic dodecahedra. It is easy to see that at most one vertex of the dodecahedron will be cut off by a ball of radius 3. There are two types of vertices: one with 3 edges, the other with 4 edges. Thus, the excess volume is less than $(V_{rd} - V_{ub})\frac{1}{7}$, where V_{rd} is the volume of a rhombic dodecahedron and V_{ub} is the volume of a unit ball. Therefore the size of any maximal independent set in G is at most $\frac{V_G}{(V_{rd}-V_{ub})\frac{6}{7}} = 11.23|mcds(G)| + 23.33$. □

Note that the bound in Theorem 6 is not a tight one. If the rhombic dodecahedra were regular polyhedra, then the bound would improve to $10.367|mcds(G)| + 21.53$. Still, this theorem shows the best bound established so far.

In a recent paper, Huang et al. [53] present a $(10 + \epsilon)-$approximation algorithm to find the minimum-weight connected dominating set problem in unit disk graphs with vertex weights.

Another type of domination problem is the m-connected k-dominating set problem (m-k-CDS). In this case, the dominating set is required to be m-connected and every vertex of the graph not in the dominating set is required to have at least k

neighbors in the dominating set. Shang et al. [73] present algorithms with performance ratios $(5 + \frac{5}{k})$ if $k \leq 5$ and 7 if $k > 5$ for 1-k-CDS problem, and $(5 + \frac{25}{k})$ if $2 \leq k \leq 5$ and 11 if $k > 5$ for 2-k-CDS problem. Thai et al. [76] present an approximation algorithm for the general m-k-CDS problem. The main idea used in these approximation frameworks is to construct a dominating set by adding maximal independent sets k times and meanwhile adding further nodes at each iteration to ensure m-connectivity. If $k \geq m$, the approximation ratio is $(8.609 + k)(2k - 1)$, otherwise we have $(8.609 + m)(2k - 1)$. These algorithms all work for UBGs as well. Based on Lemma 1, the algorithms by Shang et al. [73] have ratios $(12 + \frac{12}{k})$ for $k \leq 12$ and 14 if $k > 12$ for 1-k-CDS problem and $(12 + \frac{25}{k})$ if $2 \leq k \leq$, and 11 if $k > 5$ for 2-k-CDS problem on UBGs. Similarly, the algorithm of Thai et al. [76] in UBGs has a performance ratio of $(16.4849 + max(k, m))(2k - 1)$.

In a recent paper, Zhang et al. [85] propose a PTAS for the minimum connected dominating set in UBGs. They first state that the PTAS for the same problem in UDGs presented by Cheng et al. [26] cannot be extended to UBGs. Furthermore, when their method is applied on a UDG, the running time is improved. Next we sketch their algorithm, which is based on shifting and partitioning.

Let Q denote the minimal 3-dimensional cube containing all the unit balls with an edge length of q. Form $Q' = \{(x, y, z) | -m \leq x \leq mp, -m \leq y \leq mp, -m \leq z \leq mp\}$, where $m = \lceil 300\rho/\epsilon \rceil$ for a given $\epsilon < 1$ and a constant approximation ratio ρ for minimum connected dominating set problem on UBG. Divide Q' into $(p+1) \times (p+1) \times (p+1)$ grid such that each cell is an $m \times m \times m$ cube. Let $P(0)$ denote this partition. By shifting $P(0)$ by a units in all three coordinates, another partition $P(a)$ is obtained for $a = 0, 1, 2, \ldots, m-1$. Further, for each cell of the grid a boundary region and a central region is determined. Given the geometrical representation, algorithm starts by running a ρ-approximation algorithm for the minimum connected

dominating set. Suppose the solution of this approximation is the set D_0. Next, the partition $P(a^*)$ with the minimum number of boundary region vertices in the set D_0 is chosen. For each cell e of $P(a^*)$, consider the subgraph G_e induced by the central region vertices of the cell e. The next step is to compute a minimum subset D_e of vertices in the central region of e such that for each component H of G_e, $G[D_e]$ has a connected component dominating H. The algorithm outputs the union of all sets D_e for each cell e in $P(a^*)$ together with the union of the boundary region vertices of $P(a^*)$ in D_0. The running time of this algorithm is $n^{O(1/\epsilon^3)}$.

III.3. Chromatic Number

Recall that a *proper coloring* of a graph is one in which every vertex is colored such that no two vertices of the same color are adjacent. A graph is said to be *k-colorable* if it admits a proper coloring with k colors. The graph coloring problem is to find a proper coloring with the least number of colors, which is called the *chromatic number* of the graph.

Clark et al. [29] proved the NP-hardness of the graph coloring problem in unit disk graphs. They show that deciding whether a unit disk graph with given representation can be colored with three colors is NP-complete by using a reduction from 3-colorability of planar graphs with maximum degree 3. A direct implication of this result is that we cannot have an approximation algorithm for the coloring problem in unit disk graphs with a performance ratio smaller than 4/3 [36], unless P=NP. Based on this, we cannot expect to have an approximation scheme with a smaller performance ratio for the same problem in unit ball graphs as well.

Theorem 7. *Let $G = (V, E)$ be a unit ball graph. There exists an $O(|V| + |E|)$ algorithm for the graph coloring problem on G which has a performance ratio of at*

most 8.

Proof. This is an extension to the 3-approximation algorithm for coloring in unit disk graphs [66].

Let $deg(v)$ denote the degree of a vertex v and $\delta(G)$ denote the minimum vertex degree in a graph G. Define $\sigma(G) = \{\max\ \delta(H) |\ H$ is a subgraph of $G\}$. Szekeres and Wilf [74] proved that every graph G can be colored using $\sigma(G) + 1$ colors. Hochbaum [51] describes a way of finding such a coloring and evaluating $\sigma(G)$ in $O(|V| + |E|)$:

Step 0: Set $\sigma = 0$.

Step 1: If G has no vertices left then stop; otherwise choose a vertex v of smallest degree.

Step 2: Set $\sigma = \max\{\sigma, deg(v)\}$. Remove v and edges incident to v from G and return to Step 1.

Let v_i denote the vertex removed from G in the i^{th} iteration. Then v_i has at most σ neighbors among $v_{i+1}, v_{i+n}, \ldots, v_{i+n}$. Color vertices starting from v_n to v_1 by assigning the smallest integer that has not been assigned to its neighbors. Thus we have a $(\sigma(G) + 1)$-coloring.

Now let G be a unit ball graph and H^* be the subgraph with $\delta(H^*) = \sigma(G)$. Let v^* be the vertex in H^* with the left most X-coordinate in the unit ball representation. By Lemma 2, the subgraph induced by the vertices in $N(H^*)$ has an independent set of size at most 8. Thus in any valid coloring of this induced subgraph, no more than 8 vertices can belong to the same color class. Therefore any valid coloring of the subgraph induced by $N(H^*) \cup \{v^*\}$ must have at least $|N(H^*)|/8 + 1$ colors. So,

$$\chi(G) \geq |N(H^*)|/8 + 1$$

and by construction:

$$|N(H^*)| \geq \sigma(G).$$

Hence we have:

$$\chi(G) \geq \sigma(G)/8 + 1.$$

By the above algorithm outlined by Hochbaum [51], we can find a $(\sigma(G)+1)$-coloring of G. Thus the performance guarantee of this algorithm in unit ball graphs is 8. □

Lemma 10. *Any triangle-free unit ball graph can be colored using 9 colors.*

Proof. This is an extension to Lemma 4.1 in [66]. Every triangle-free unit ball graph has a vertex with degree at most 8. Let G be a triangle-free unit ball graph. Since G is triangle-free, the neighborhood of each vertex in G is an independent set. Thus, from Lemma 2, there exists a vertex whose degree is at most 8. Consider a simple algorithm which removes a vertex with degree at most 8 at each iteration. Let v_i denote the vertex removed at i^{th} iteration. Thus, we have an ordering of vertices $\{v_1, v_2, \ldots, v_n\}$. We can color all the vertices starting from v_n to v_1 with at most 9 colors. Consider the coloring of v_i. Since it has at most 8 neighbors in $\{v_{i+1}, \ldots, v_n\}$, we can use the 9^{th} color that is not used by any of its neighbors. This is true for any i. □

III.4. Max-cut

Let $G(V, E)$ denote a graph. A *cut* is a partition of the vertices V into two subsets V_1 and V_2. Any edge $(u, v) \in E$ with $u \in V_1$ and $v \in V_2$ is said to be crossing the cut and is a *cut edge*. The *size of a cut* is the total number of edges crossing the cut. In edge-weighted graphs, the size of the cut is defined to be sum of weights of

the edges crossing the cut. The *max-cut problem* is to find a cut of maximum size. The variation, in which each partition set is required to have the same cardinality, is called the *max-bisection problem*.

In a recent paper, Diaz et al. [32] prove that both the max-cut and the max-bisection problems are NP-hard on unit disk graphs. They use the fact that max-cut problem is NP-hard on graphs with bounded degree for $\Delta \geq 3$. They show that max-cut on a graph with bounded degree 4 can be reduced to the max-cut problem on a unit disk graph in polynomial time. The complexity proof of max-bisection easily follows by a reduction from max-cut in unit disk graphs. More specifically, finding the max-cut in G is equivalent to finding a max-bisection in G', which is formed by two copies of G. Hunt el al. [54] investigate PTAS's for various problems in unit disk graphs. They state that there is a PTAS for max-cut problem in λ-precision unit disk graphs (λ-precision implies that centers of unit disks are at least distance λ away from each other).

Jansen et al. [55] present a PTAS for max-bisection in unit disk graphs. By imposing grids on the plane, the graph is divided into subgraphs. If the subgraph is dense, then PTAS for dense instances of max-bisection is used as proposed by Arora et al. [9]. Otherwise, the solution is obtained by enumeration. Authors also use shifting to obtain different subgraphs. The solution for each subgraph S, all maximum $(n_s - p, p)$ partitions, where n_s is the number of vertices in S, are used to obtain an overall solution for the input graph via dynamic programming. We observe that this technique can also be used for unit ball graphs with some adjustments.

Theorem 8. *Max-bisection and max-cut problems both admit PTAS in unit ball graphs.*

Now we give a sketch of the algorithm of Jansen et al. [55] with the necessary

adjustments.

Obtaining subgraphs: The first step is to impose a 3-dimensional grid. Each cell of this grid is a cube with side length 2. The h-th yz-plane is at $x = 2h$, $-\infty < h < \infty$. Furthermore, the h-th yz-strip is the strip between the h-th and $(h+1)$-st yz planes. The xz-planes and xz-strips, xy-planes and xy-strips are indexed similarly. Each strip is open on one side and closed on the other side. Thus, each unit ball is centered in exactly one strip.

For a fixed integer k, the subgraph $H_{i,j,l}$ of the input graph G, $-\infty < i, j, l < \infty$, is the subgraph induced by the centers of unit balls that lie in the intersection of the yz-strips $i, i+1, \ldots, i+k$, the xz-strips $j, j+1, \ldots, j+k$ and the xy-strips $l, l+1, \ldots, l+k$. The number of vertices of $H_{i,j,l}$ is denoted by $n_{i,j,l}$. We observe that the size of a maximum independent set on $H_{i,j,l}$ is bounded by $6(k+2)^3/\pi$. The unit balls in $H_{i,j,l}$ are contained in a cube with side length $2(k+1)+2 = 2(k+2)$. Dividing the volume of this cube by the volume of a unit ball, we can fit at most $6(k+2)^3/\pi$ non-adjacent unit balls in this region.

Lemma 4 of Jansen et al. [55] also holds for subgraph $H_{i,j,l}$. The lemma states that there is a positive constant c such that if $n_{i,j,l} > c\log(n)$, then the subgraph $H_{i,j,l}$ is dense. This is proved by showing that there exist at least $n_{i,j,l}\pi/6(k+2)^3$ maximal independent sets in $H_{i,j,l}$. If each maximal independent set is considered as a vertex, since they are all maximal, we have a complete graph. Thus, the number of edges in $H_{i,j,l}$ is $\Omega((n_{i,j,l})^2)$. Thus, Corollary 4 of the same paper [55] also holds. This corollary states that if $n_{i,j,l} > c\log(n)$, then the size of a max-bisection of $H_{i,j,l}$ is $\Omega((n_{i,j,l})^2)$. This directly shows that the max-cut size of $H_{i,j,l}$ is also $\Omega((n_{i,j,l})^2)$.

Solution for each subgraph: For each i, j, l, we compute all maximum $(n_{i,j,l}-p, p)$

partitions of $H_{i,j,l}$. If $H_{i,j,l}$ is dense, a solution with an additive error of $2\epsilon(n_{i,j,l})^2$ is obtained by solving a polynomial integer program, which is presented by Jansen et al. [55] and Arora et al. [9]. Otherwise, an optimal solution is computed by enumeration.

Combining solutions to get an overall solution: The graph $G_{r,s,t}$ for each r, s and t, $0 \leq r, s, t \leq k$, is defined as the union of all subgraphs $H_{i,j,l}$, where $i(modk+1) = r$, $j(modk+1) = s$ and $l(modk+1) = t$. All maximum $(n_{r,s,t} - p, p)$-partitions of $G_{r,s,t}$ are obtained by merging solutions of each subgraph $H_{i,j,l}$ of $G_{r,s,t}$. First, the subgraphs are ordered in increasing order of the sum $i+j+l$. All partitions of consecutive pairs are computed first by using solutions of each subgraph, then the same is done for quadruples by using the solutions of pairs. In this fashion, all maximum partitions are computed for $G_{r,s,t}$.

Output: For max-bisection, we pick the largest bisection of $G_{r,s,t}$, $0 \leq r, s, t \leq k$, and for max-cut we pick the largest partition.

Lemma 11. *The performance ratio of the above algorithm is* $(1 - \frac{1}{k+1})^3(1 - 2\epsilon)$ *for a fixed integer* k*, and a given* $\epsilon < 1$*.*

Proof. The proof is inline with the two-dimensional case [55]. Suppose we deal with max-cut problem. Let C^* be a max-cut of G. Observe that any graph $G_{r,s,t}$ misses some edges that are present in G. Particularly, those are the edges that have endpoints in different subgraphs $H_{i,j,l}$ of $G_{r,s,t}$. For any edge e of C^*, there is at most one r, $0 \leq r \leq k$, such that e cuts a yz-plane whose index modulo $k+1$ is r for fixed s and t. The same holds for any fixed pair of (r, s, t). Thus, there exist a graph $G_{r,s,t}$, $0 \leq r, s, t \leq k$, such that a max-cut of $G_{r,s,t}$ has at least $(1 - \frac{1}{k+1})^3|C^*|$ edges. Furthermore, since the dense subgraph solutions have an additive error of $2\epsilon(n_{i,j,l})^2$

and each such subgraph has a max-cut of $\Omega((n_{i,j,l})^2)$, the solution obtained for each $G_{r,s,t}$ is a $(1-2\epsilon)$-approximation of the optimal solution. Therefore, for the input graph G, we obtain a performance ratio of $(1-\frac{1}{k+1})^3(1-2\epsilon)$. □

III.5. Min-bisection

A partition of vertices of a graph G into two equal-cardinality sets is called a *bisection*. The *min-bisection problem* is to find a bisection with minimum number of edges with endpoints in different partition sets. The computational complexity of this problem is unknown for unit disk graphs and unit ball graphs.

Theorem 9 ([33])**.** *If min-bisection is NP-complete for planar graphs with maximum vertex degree 4, then it is NP-complete even when restricted to unit disk graphs.*

However, complexity of min-bisection for planar graphs, and planar graphs with bounded degree has not yet been proven.

Minimum weighted bisection problem: Given a vertex-weighted graph G, partition the vertices of G into two sets such that the total weight of each set is equal and the number of the edges in between these partitions is minimized.

Partition problem: Given a multi set S of integers, is there a way to partition S into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2?

Partition problem is a well-known NP-complete problem.

Theorem 10. *Minimum weighted bisection is NP-hard in unit disk graphs.*

Proof. We apply a reduction from the partition problem. Given a multi set $S = \{a_1, a_2, \ldots, a_k\}$ of integers, we form a graph G in the following way: assign a vertex

for each element a_i of S such that the weight of the vertex is a_i. Place each vertex on the plane so that their distances are always greater than 1. Thus we have a weighted unit disk graph G which does not have any edges.

S has a partition $(S1, S2)$ with equal sums if and only if G has a bisection of size 0. So, if we were able to find the minimum weighted bisection on G in polynomial time, then we would conclude we can answer the partition problem in polynomial time too. Since the partition problem is NP-complete, the minimum weighted bisection in unit disk graphs is NP-hard.

□

The computational complexity and approximation status of min-bisection problem in unit disk and unit ball graphs are open problems.

III.6. Maximum Independent Set Problem

An *independent set* of a graph G is a subset of mutually non-adjacent vertices. The *maximum independent set problem* is to find an independent set of maximum cardinality.

Clark et al. [29] prove NP-hardness of the maximum independent set problem on unit disk graphs. Cerioli et al. [23] show that it is NP-hard even for penny graphs (non-overlapping unit disk graphs).

Theorem 11 ([51])**.** *It takes only $O(nlogn + m)$ steps to find in any weighted graph G with n vertices, m edges and no $(p+1)$-claw a stable set whose weight is at least $1/p$ times the weight of an optimal stable set.*

A unit ball graph is a $12-$claw free graph by Lemma 1. Thus, by Theorem 11 there is an 11-approximation algorithm for maximum independent set problem on unit ball graphs.

Marathe et al. [66] give a 3-approximation algorithm for unit disk graphs. This can be easily extended to an 8-approximation algorithm for unit ball graphs:

Theorem 12. *There exists an* 8*-approximation algorithm for finding the maximum independent set in a unit ball graph.*

Proof. Let G be a unit ball graph. By Lemma 2, there exists a vertex v in G, whose neighborhood contains an independent set of size 8. If the geometrical representation is given, finding such a vertex is easy. If this information is not available, then such a vertex can be found in polynomial time by checking the neighborhood of every vertex. Each time such a vertex is found, it is added to the independent set and removed from the graph together with its neighbors. The next search is done on the remaining graph. This way we obtain an independent set S. Observe that by construction each vertex in $V(G) - S$ is a neighbor to at least one vertex in S. Furthermore, each time a vertex is added to S, we know its neighborhood in the remaining graph forms an independent set of size at most 8. Thus, it is easy to see that maximum independent set size can be at most $8|S|$. □

Matsui [67] presents an approximation algorithm for the maximum independent set in unit disk graphs. The author considers unit disk graphs defined on a slab that has a width of k (meaning that the y-coordinate of the point set is contained in $[0, k)$). It is showed that if $k < \sqrt{3}$, the problem can be solved in polynomial time ($O(n^2)$) since the graph becomes a co-comparability graph and the problem reduces to a longest path problem. For a fixed width k, by using similar ideas, a polynomial time algorithm is achieved with a running time of $O(n^{4\lceil k/\sqrt{3}\rceil})$. This is extended to a $(1 - 1/r)-$approximation algorithm for the maximum independent set problem on general unit disk graphs with a running time of $O(rn^{4\lceil (r-1)/\sqrt{3}\rceil})$.

Halldórsson [47] uses local improvement search techniques to obtain an NC algorithm with an approximation ratio of $\frac{k}{2}+\varepsilon$ for the maximum independent set problem restricted to $(k+1)-$claw free graphs for any $k \geq 4$. Since a unit ball graph is $12-$claw free, this NC algorithm has a performance ratio of $5.5+\varepsilon$.

When the unit disk representation of a graph is given, polynomial time approximation schemes exist for the maximum independent set problem in unit disk graphs mostly using a technique called shifting. Such an algorithm is presented by Hunt et al. [54] and we observe that it can be easily extended to unit ball graphs. A sketch of the algorithm for unit disk graphs is presented by Erlebach and Fiala [36]. Now we give the extended version of this sketch for unit ball graphs with some adjustments.

Let B be a set of unit balls and I^* denote an optimal independent set. To overcome the scale difference with Erlebach and Fiala's algorithm [36], we scale the graph by multiplying each center with $\frac{1}{2}$. We assume that scaled centers of the unit balls do not have integral coordinates. Next we consider the $3-$dimensional grid which is composed of xy, yz and xz planes at all integer coordinates. An integer $p > 0$ is fixed. Let $B_{i,j,k}$ denote the subset of unit balls obtained by removing all balls that intersect any of the following planes for some integer t: $x = i + pt$, $y = j + pt$, $z = k + pt$. Now we consider the cubes obtained by removal of the above planes. It is easy to see that each unit ball in $B_{i,j,k}$ is completely contained in a cube. Thus we can find the maximum independent set of $B_{i,j,k}$ by combining the maximum independent sets of all cubes. Since in a cube we can have an independent set of size at most $O(p^3)$, maximum independent set for each cube can be computed in time $|B|^{O(p^3)}$ by enumeration. Repeating this for all subsets such that $0 \leq i, j, k \leq p-1$, we output the largest solution I_p. Observe that the number of the repetitions is p^3.

Now we show that $|I_p| \geq (1-\frac{3}{p})|I^*|$. Observe that a unit ball in B can intersect at most one $xz-$plane , at most one $yz-$ plane and at most one $xy-$ plane. Thus

for a given subset $B_{i,j,k}$, at most $|I^*|/p$ balls intersect $xy-$ plane ($yz-$ plane, $xz-$ plane). Hence $|I_p| \geq (1 - \frac{3}{p})|I^*|$.

So, in order to have a $(1-\varepsilon)-$algorithm, we can choose $p = \lceil 3/\varepsilon \rceil$. The algorithm has a running time of $|B|^{O(p^3)}$.

III.7. Minimum Vertex Cover

A vertex cover for an undirected graph G is a subset S of its vertices such that each edge has at least one endpoint in S. The minimum vertex cover problem is to find a vertex cover of minimum cardinality.

Theorem 13 ([51, 66])**.** *Let G be a weighted graph with n vertices and m edges; let k be an integer greater than one. If it takes s steps to color the vertices in G in k colors, then it takes only $s+O(nm \log n)$ steps to find a stable set whose weight is at least $2/k$ times the weight of an optimal stable set and to find a cover whose weight is at most $2 - 2/k$ times the weight of an optimal vertex cover.*

Lemma 12 ([66, 14])**.** *Let r_1 and r_2 denote the local ratios of two heuristics for the vertex cover problem. If G_1 denotes the graph obtained after applying the first heuristic to a graph G, then the performance of the algorithm which applies the two heuristics in succession is at most $max\{r_1(G), r_2(G_1)\}$.*

Theorem 14. *There exists an $\frac{16}{9}-$approximate algorithm for the minimum vertex cover problem in unit ball graphs.*

Proof. This is an extension to a 1.5-approximation algorithm for Vertex Cover in unit disk graphs [66].

Let G be a unit ball graph. First remove the vertices that form a triangle and include them in the vertex cover, till the graph becomes triangle-free. The remaining

graph is 9-colorable by Lemma 10. Thus, by Theorem 13 there exists an algorithm that finds a vertex cover on the remaining graph, which is at most 16/9 times the weight of the optimal. The local ratio of the first part, where vertices forming triangles are removed, is 3/2, since we pick all 3 vertices, where 2 vertices of any triangle have to belong to the optimal vertex cover. Hence, by Lemma 12, the performance is at most $\max\{3/2, 16/9\} = 16/9$. □

The minimum vertex cover problem has a PTAS when the input graph is a unit disk graph [54]. We observe that by using the method presented for the PTAS for the maximum independent set problem in the previous section, we can obtain a PTAS for the minimum vertex cover problem in unit ball graphs. The only difference is that for the minimum vertex cover problem instead of removing the unit balls that intersect with the $xy-$, $yz-$ and $xz-$ planes, we count them twice for any given plane. More specifically, when we consider the subgraphs for which the optimal solution is computed for the maximum independent set problem, they are composed of the unit balls that completely lie in a given cube (corresponding a subgraph). For the minimum vertex cover problem, we consider all the unit balls that lie either partially or completely. Again, for each subgraph the optimal solution can be found in polynomial time since the minimum vertex cover $VC(G)$ of a graph G is given by $VC(G) = V(G) - IS(G)$, where $V(G)$ and $IS(G)$ denote the vertices of G and the vertices of G in a maximum independent set of G, respectively.

Let V^* denote an optimal vertex cover of a unit ball graph G. Let V^p denote the output of the algorithm for a given p. Similar to the observation that we had for the previous section, we observe that a unit ball in G can intersect at most one $xz-$plane, at most one $yz-$ plane and at most one $xy-$ plane. Thus for a given subset $B_{i,j,k}$, at most $|V^*|/p$ balls intersect $xy-$ plane ($yz-$ plane, $xz-$ plane) meaning that at most

$|V^*|/p$ balls are counted twice for $xy-$ plane ($yz-$ plane, $xz-$ plane). Therefore, $|V^p| \leq (1 + \frac{3}{p})|V^*|$. Choosing $p = \lceil 3/\varepsilon \rceil$, we have a $(1 - \varepsilon)-$algorithm.

III.8. Maximum Clique Problem

Given an undirected graph G, a *clique* is a subset of pairwise adjacent vertices in G. The maximum clique problem is to find a clique of the largest cardinality in G. The size of a maximum clique is called the clique number of G and is denoted by $\omega(G)$).

Most NP-complete problems in general graphs preserve their hardness in unit disk graphs. At first glance, it is surprising to see that one of the most famous NP-hard problems on general graphs, the maximum clique problem, can be solved in polynomial time when restricted to unit disk graphs [29]. Given any pair of adjacent vertices, we observe that the intersection of the neighborhoods within the distance in between members of the pair, forms a special induced subgraph. Actually, the complement of this subgraph is a bipartite graph, in which the maximum independent set problem can be solved in polynomial time. Thus, the maximum clique on this subgraph can be determined in polynomial time. Moreover, there exists a pair of adjacent vertices that contains all the vertices of the maximum clique. This leads to a polynomial time algorithm for the whole graph by finding the maximum clique on subgraphs of every adjacent pair of vertices and picking the largest one.

The complexity for the unit ball graphs is not yet known. Afshani and Hatami [3] prove the NP-hardness of this problem in higher dimensions.

Theorem 15 ([3]). *For every $\varepsilon > 0$ there exists $d = \Theta(logn)$, such that there is no polynomial time $(\frac{95}{94} - \varepsilon, \sqrt{\frac{4}{3}} - \varepsilon)$- approximation for the maximum clique problem in dimension d, unless P=NP.*

The authors apply a reduction from the maximum independent set problem in

3-regular graphs. By using eigenvalues of the Laplacian of the complement of the 3−regular graph, a representation of the vertices in Euclidian space $\mathbb{R}^n$ is obtained such that adjacent vertices are at a specified distance. The distance between non-adjacent vertices is also specified, and it is greater than that of the adjacent ones'. Next they use dimension reduction techniques and show that there exists a dimension $d = O(\lambda^{-2} log n)$ such that the vertices can be mapped into $\mathbb{R}^d$ in polynomial time and the distances between vertices change by a factor at most $1 + \lambda/2$.

So, Theorem 15 does not specifically answer the question of complexity in dimension 3.

Afshani and Chan [2] introduce approximation algorithms for the maximum clique problem in unit ball graphs. The main idea in these algorithms is to cover the intersection of the neighborhoods of adjacent vertices with a small number of shapes of diameter at most one. If this number is k, then the complement of the subgraph induced by the vertices in this intersection forms a $k-$partite graph. Taking any pair of regions, maximum clique on the subgraph induced by the vertices in those regions can be found in polynomial time. Repeating this for every possible pair and choosing the best, we have an approximation for the maximum clique. The authors achieve their best ratio of 2.553 by using a shape which they name as "rounded diamond". They show that they can cover the intersection area by 5.106 such rounded diamonds on average. The running time of this algorithm is $O(n^3 T_{match}(n))$, where $T_{match}(n)$ is the running time of matching algorithm for n vertices. They also show that a 5.106− approximation is achieved with a running time of $O(n \log n)$ without using matching.

The computational complexity and the existence of a PTAS for the maximum clique problem in unit ball graphs are interesting open problems.

III.9. Minimum Clique Partition

The *minimum clique partition problem* is to partition a given graph G into a minimum number of cliques. This problem is proven to be NP-complete for unit disk graphs by Cerioli et al. [23]. The authors also present a 3-approximation algorithm with a running time of $O(n + \overline{m})$, where $\overline{m}$ denotes the number of edges in the complement of the given graph. As stated earlier, a unit disk graph is called a k-strip or k-slab if the y-coordinates of all the centers of unit disks is contained in the interval $[0, k)$. The authors state that when $k \leq \sqrt{3}$, the graph becomes a co-comparability graph and the minimum clique partition can be found by coloring its complement in time $O(n+\overline{m})$. Given a general unit disk graph, it is partitioned into $\sqrt{3}$-strips and the problem is solved optimally for each strip. The union of all the solutions is an approximation for the input graph. In order to show the approximation ratio of 3, they use the fact that any vertex in a clique belongs to at most 3 strips. The authors also show that for non-overlapping unit disk graphs, an approximation ratio of 3/2 can be achieved in $O(n \log n)$ time.

Theorem 16. *There exists a* 20*-approximation algorithm for minimum clique partition in unit ball graphs.*

Proof. Let G be a unit ball graph. Consider the following algorithm.

Step 1: Decompose G into $(\sqrt{2}, 1)-$ slabs.

Step 2: For each slab, find an exact clique partition on the subgraph induced by the vertices of G who are located in the given slab.

Step 3: Output the union of solutions from Step 2.

The above algorithm is an extension to the one presented by Cerioli et al. [23] for unit disk graphs. By Theorem 1, each subgraph in Step 2 is a co-comparability graph. For

each subgraph, a minimum clique partition can be found by coloring the complement. Thus, Step 2 of the algorithm can be performed in time $O(n + \overline{m})$, where $\overline{m}$ denotes the number of edges of the complement of G.

The approximation ratio can be proven by a geometrical argument that the vertices of a clique can be contained in a region distributed along at most 20 $(\sqrt{2}, 1)$-slabs as shown in Figure 8.

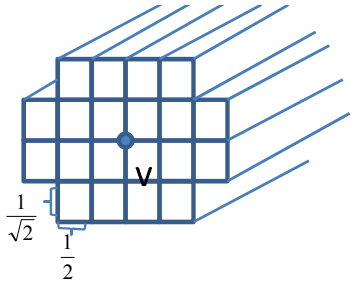

Fig. 8 The cliques that cover vertex v can be contained in a region spanned by the above 20 $(\sqrt{2}, 1)-$slabs.

□

Whether there exists a PTAS for minimum clique partition problem in both unit disk and unit ball graphs is an interesting open problem.

III.10. Open Problems

Next we give a list of open problems regarding optimization problems in unit disk and unit ball graphs. Although unit disk graphs have been studied for a long time, the complexity and approximation status of several well-known optimization problems are still open problems, including the following problems:

1. The computational complexity and approximation status of min-bisection problem in unit disk and unit ball graphs.

2. The computational complexity and existence of a PTAS for the maximum clique problem in unit ball graphs.

3. Existence of a PTAS for minimum clique partition problem on both unit disk and unit ball graphs.

III.11. Conclusion

In this chapter, we presented a survey of complexity and approximation algorithms for several optimization problems restricted to unit ball graphs. Unit disk graphs, which are the two-dimensional version of the unit ball graphs, have been studied in the literature for many years. Yet, complexity and approximation status of several optimization problems, such as minimum bisection and minimum clique partition, are still open problems in unit disk graphs. We worked on the extension of the existing approximation algorithms for unit disk graphs to three-dimensional space and the analysis of computational complexity since UBGs provide a more realistic representation of wireless networks. We observe that several approximation algorithms for unit disk graphs can be easily extended to unit ball graphs and we provide the corresponding performance guarantees. We highlight several interesting open problems, such as the computational complexity of the well-known maximum clique problem restricted to unit ball graphs. This problem is polynomially solvable in unit disk graphs. Furthermore, we propose and prove some properties of unit disk and unit ball graphs which can be used in analyzing algorithms.

CHAPTER IV

THE MAXIMUM K-CLIQUE PROBLEM IN UDGs

The maximum clique problem is a very well-known optimization problem. It is most famous in the social network context, where one is interested in finding a cohesive subgroup. Due to the criticisms for its overly restrictive nature, several clique relaxations have been introduced. These include diameter relaxations (k-club), degree-based relaxations (k-plex) and distance relaxations (k-clique). In this work, we are interested in the maximum k-clique problem when the graph is restricted to be a unit disk graph. Recall that a unit disk graph is a typical graph model used to describe wireless networks. In order to avoid the interference between different wireless nodes when they broadcast at the same time, it is important to assign different frequencies to the nodes whose transmission ranges intersect. Thus, the clique number gives the least number of frequencies needed. Fortunately, the clique number of a unit disk graph can be computed in polynomial time [29]. However, since the unit disk graph model is only a rough approximation of the wireless network topology, clique relaxations may provide a better estimate of the number of frequencies needed.

Given a graph G, a subset of vertices S of G is a *k-clique* if the distance between any two vertices from S in G is at most k. The *maximum k-clique problem* is to find a k-clique of maximum cardinality. It is important to note that in the literature this notation has also been used for a clique of cardinality k. The definition we used is a common definition in social networks. Moreover, in some papers it is referred to as *distance-k clique*.

Balasundaram et al. [13] proved that for any fixed positive integer k the maximum k-clique problem is NP-hard in general graphs. The proof is done by a reduction

from the maximum clique problem. Since the maximum clique problem is polynomially solvable in unit disk graphs, we cannot use the same argument to show the complexity of maximum k-clique in unit disk graphs. Another important remark is that the authors of [13] state that this problem is hard to solve not only because it is a generalization of the maximum clique problem, but because it is hard in its own respect. Thus, although the maximum clique problem is polynomial-time solvable for unit disk graphs, the maximum k-clique problem for $k > 1$ may not be so.

IV.1. Literature Review

Although the concept of k-clique has been introduced in social network analysis literature as early as in 1950s, there are very few publications related to this problem. Balasundaram et al. [13] propose using k-cliques and k-clubs as an alternative approach to finding clusters in biological networks. The authors also present the first computational complexity results. For general graphs and for graphs with bounded diameter, the maximum k-clique problem is NP-hard. Wu et al. [80] investigate the minimum edge density of a 2-clique in a directed graph.

A related optimization problem is the minimum k-clique partition. Edachery et al. [35] demonstrate implementation of several clustering algorithms based on k-cliques.

There is no work done in the literature when the graph is restricted to be a unit disk graph.

IV.2. Computational Complexity

The maximum k-clique problem in a graph G is equivalent to the maximum clique problem in G^k. By Lemma 7 of Chapter III, G^k is not necessarily a unit disk graph.

There are some strong indications that this problem is NP-hard in unit disk graphs. We claim this as a conjecture and give complexity results for some geometric graphs which may be helpful in verifying our conjecture.

For $\epsilon \in [0, 1]$, a *unit ϵ-quasi-disk* is a connected compact set Q of the plane such that there exists a point P such that $D(P, 1-\epsilon) \subseteq Q \subseteq D(P, 1)$, where $D(C, r)$ denotes the disk of radius r centered at C [24]. Figure 9 illustrates this concept.

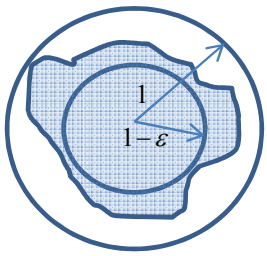

Fig. 9 A unit ϵ-quasi-disk.

Ceroi [24] proved that the maximum clique problem on the class of intersection graphs of unit *ϵ-quasi-disks* is NP-hard for any $\epsilon > 0$. The proof is done by a reduction from the maximum independent set problem on cubic graphs.

Lemma 13. *Given a unit disk graph G, G^k is an intersection graph of* unit ϵ-quasi-disks *with* $\epsilon = \frac{k-1}{k}$.

Proof. Recall that the intersection model, the containment model and the proximity model are all equivalent with some scaling modifications on the unit distance. It is easier to prove this lemma by considering the containment model. Given a unit disk graph G with unit distance 1 for the containment model, it is easy to see that for a given vertex v any neighbor of v in G^k is at most distance k from v in G, thus all the neighbors of v in G^k are contained in a disk of radius k. Furthermore,

there may exist vertices that are not adjacent to v but are at distance less than k, as in Figure 10. Thus, the actual neighborhood region is a subset of vertices covered by the disk of radius k. Let O denote the point that corresponds to vertex v. Since G is a unit disk graph, all vertices that are at most distance 1 from v are contained in the neighborhood region. Let Q denote the set of points, such that each point, if added as a new vertex to G, would be a neighbor of v in G^k. Then, we have $D(O,1) \subseteq Q \subseteq D(O,k)$. When we scale all the coordinates of the points by multiplying by $\frac{1}{k}$, we get $D(O,1-\epsilon) \subseteq Q \subseteq D(O,1)$ with $\epsilon = \frac{k-1}{k}$, where for convenience we use the same notations as before for transformed O and Q. Now, we need to show that Q is a connected compact set. It is compact since it is closed and is a subset of a compact set $D(O,k)$. Now, assume that Q is not connected. Let u be a vertex in a compact set Q_2 and let the vertex v belong to the compact set Q_1 such that $Q_1 \bigcup Q_2 = Q$ and $Q_1 \bigcap Q_2 = \oslash$. Let T denote the shortest path from v to u in G (see Figure 11). Then there exist vertices $t_1 \in Q_1$ and $t_2 \in Q_2$ in T such that they are adjacent in G. As shown in Figure 12, consider the disk $D(t_1, dist(t_1,t_2))$. Since t_2 is contained in Q, any vertex that lies on this disk also belongs to Q as it has a shortest path to v of length less than k, since $dist(v,t_1) < dist(v,t_2) \leq k$. Therefore, we have $Q_1 \bigcup Q_2 \bigcup D(t_1, dist(t_1,t_2)) \subseteq Q$. Observe that not having any vertices of G in this disk does not violate any property. Thus, Q is connected. □

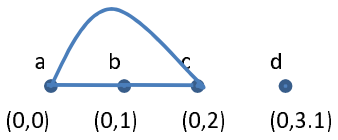

Fig. 10 k^{th} power of G for any integer $k > 0$.

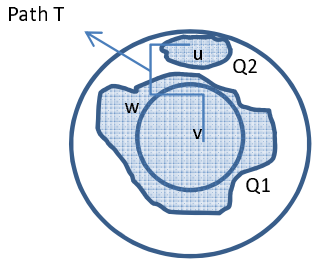

Fig. 11 $Q_1 \bigcup Q_2 = Q$ contains the k-distance neighborhood of vertex v in G for $k > 0$.

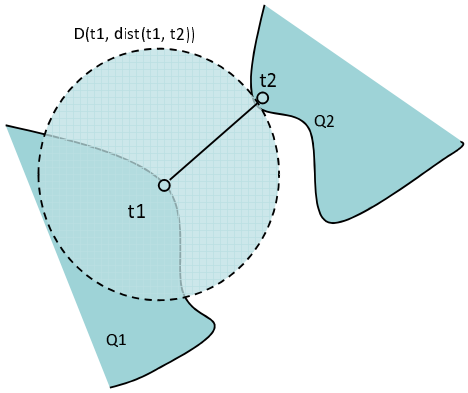

Fig. 12 Q is a connected compact set.

By Lemma 13, the power graphs of unit disk graphs form a subclass of intersection graphs of unit ϵ-quasi-disks, on which the maximum clique problem is NP-hard.

String graphs are intersection graphs of curves in the plane. Each vertex corresponds to a curve and two vertices are joined by an edge if the corresponding curves intersect. *Intersection graphs of ellipses* are defined in a similar fashion. An *interval graph* is the intersection graph of a multiset of intervals on the real line. It has one vertex for each interval in the set, and an edge between every pair of vertices corresponding to intervals that intersect. *Multiple interval graphs* are a natural generalization of interval graphs, where each vertex may have more than one interval associated with it. A *circle graph* is a graph whose vertices can be associated with chords of a circle such that two vertices are adjacent if and only if the corresponding chords in the circle intersect. A graph is *chordal* if each of its cycles of four or more nodes has a chord, which is an edge joining two nodes that are not adjacent in the cycle. An equivalent definition is that any chordless cycles have at most three nodes.

The maximum clique problem is polynomial-time solvable in interval graphs, chordal graphs and circle graphs. It is proved in [22] that the maximum clique problem is NP-hard in t-interval graphs for $t \geq 3$ by a reduction from the maximum 2-DNF satisfiability problem. Let ρ denote the ratio of the larger over the smaller radius of an ellipse. Ambuhl and Wagner [8] prove that the maximum clique problem is APX-hard in intersection graphs of ellipses for any $1 < \rho < \infty$. This means that there is a constant c such that there is no approximation algorithm with ratio better than c. Thus, there is no PTAS. The proof is done by a reduction from the MAX5OCC2SAT problem (given a boolean formula in conjunctive normal form with at most two literals per clause and at most for five occurrences of every variable, find an assignment of truth values to the variables that satisfies the maximum number of clauses). Observe that for $\rho = 1$, we have unit disk graphs for which the maximum

clique problem is polynomial-time solvable. Thus, even a small change in ρ makes a huge complexity difference. The same is true for $\rho = \infty$, in which case the graph becomes an interval graph.

Kratochvil and Kubena [60] show that the maximum clique problem on the intersection graph of convex sets in the plane is NP-hard. The reduction for this proof is done from the maximum clique problem on co-planar graphs by showing that for every planar graph, one can assign convex sets in the plane to its vertices in such a way that two of the sets are disjoint if and only if the corresponding vertices are adjacent. Although it is easy to see that the k^{th} power of a unit disk graph is not necessarily an intersection graph of convex sets, this result is important in terms of providing a general idea on more general intersection graphs.

IV.3. A Greedy Heuristic

In this section, we propose a greedy heuristic for the maximum k-clique problem, which will be used to get an initial solution for the exact solution procedure we are about to develop. An initial heuristic is helpful in decreasing the problem size. The outline of this heuristic is presented in Figure 13. The main idea is to start with a maximum degree vertex and add maximum number of vertices to the solution set at each iteration to obtain a k-clique. We present a demonstration of this heuristics in Figure 14. In this example, we find a 4-clique. First, we pick the vertex with maximum degree and its neighbors. We color the vertices that can be a candidate in terms of including their neighbors in the future as green. Red color indicates that the vertex is in the solution but not a candidate for improvement. Blue color indicates that the vertex is not a part of the current solution. At each step, we compare the number of blue neighbors of each green vertex and pick the largest one. Finally, the

set of colored vertices corresponds to a 4-clique.

Input: A unit disk graph G with the coordinates of the points; $k > 2$
Output: A k-clique C in G

Find a vertex in G of maximum degree, v_{max}

$C = \{v_{max}\} \bigcup N(v_{max})$

for all $w \in N(v_{max})$ **do**
    label w as candidate

end for

for $i = 1 : k - 2$ **do**
    Pick a candidate vertex $w \in C$ with the maximum degree in $G[V(G) \setminus C]$
    $C = C \bigcup N_{G[V(G)\setminus C]}(w)$
    Remove w from the candidate list
    Label all vertices in $N_{G[V(G)\setminus C]}(w)$ as candidate

end for

Return C

Fig. 13 Greedy heuristic for the maximum k-clique problem.

IV.4. An Exact Solution Procedure

Let G be a UDG with unit distance d. We define the graph G_k on the same set of vertices of G by introducing an edge in between two vertices whenever their distance is less than or equal to kd. Thus, by definition G_k is a UDG. Furthermore, we observe that G^k is a subgraph of G_k.

Since G_k is a unit disk graph, the maximum clique problem can be solved in polynomial time using matching. As proven by Clark et al. [29], the intersection area of the neighbors of a pair (u, v) of adjacent vertices within a distance at most $d(u, v)$ to both u and v form a special graph, where $d(u, v)$ denotes the pairwise Euclidian

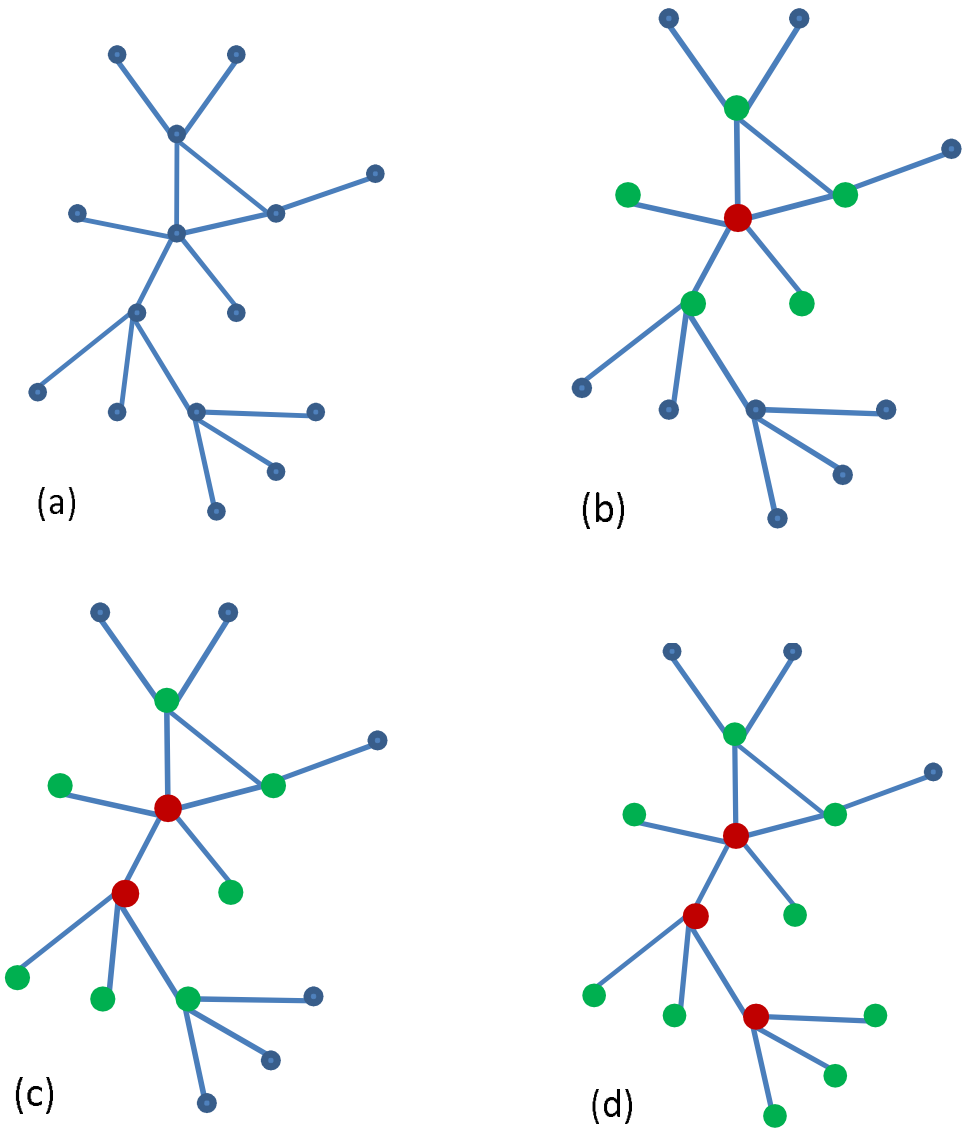

Fig. 14 Demonstration of the proposed greedy heuristic for the maximum k-clique problem for k=4 (a) Input graph G (b) Initial step: Finding the maximum degree vertex (c) Step 1: Improving the solution by adding the blue neighbors of the vertex with maximum number of blue neighbors (d) Step 2: Last step to get a 4-clique.

distance between u and v. This graph is a co-bipartite graph. The maximum clique problem in a graph is equivalent to the maximum independent set problem in the complement graph. On bipartite graphs, the maximum independent set problem can be solved in polynomial time. Clark et al. [29] also prove that there exists a pair of adjacent vertices for which the corresponding area contains the maximum clique. Thus, computing the maximum clique for each pair and picking the largest one gives an optimal solution.

The main idea of our algorithm is to solve the maximum clique problem in G_k, and do branching on some variables if the found solution is not feasible for the power graph. This is done for all pairs of adjacent vertices in G^k. Although G^k is not necessarily a unit disk graph, it is easy to see that there exists a pair of adjacent vertices for which the intersection region mentioned above contains the optimal solution. This is true since G^k is a subgraph of G_k with the same set of vertices but some missing edges.

Let G be a UDG and k be a given number. Let C^* denote the optimal solution and C_0 be the initial solution obtained by the greedy heuristic introduced in the previous section. Figure 15 gives a sketch of the algorithm. The first step is to form the graphs G^k and G_k and label the edges that belong to G_k but not to G^k as *fake*. After we obtain an initial solution C_0, we can reduce the size of the problem by eliminating the vertices whose degree in G^k is less than $|C_0 - 1|$, since such a vertex cannot be a part of a better than current solution.

The next step is the main step of the algorithm, where we iterate through all adjacent pairs to compute the maximum clique in G^k. It is important to note that this step can be improved by considering only adjacent pairs of vertices in G^k for which the distance in between is greater than $k-1$. Of course, in order to do that, we need to make sure that the diameter of each connected component of the input graph G is

Input: A unit disk graph G with the coordinates of the points, k
Output: A maximum k-clique in G

Form graphs G^k and G_k

Label $E(G_k) \setminus E(G^k)$ as fake

Obtain C_0, Set $C^* \leftarrow C_0$

for $v \in G^k$ **do**
  if $deg_{G^k}(v) < |C_0 - 1|$
    delete v from both G^k and G_k

end for

for each pair of adjacent vertices (u, v) of G^k **do**
  $S \leftarrow \{w \in V(G) : d(w, u) \leq d(u, v)$ and $d(w, v) \leq d(u, v)\}$
  for $w \in S$ **do**
    if $deg_{G^k[S]}(w) < |C^* - 1|$
    delete w from S
  end for
  if $|S| > |C^*|$
    Solve maximum clique in $G_k[S]$ using matching
    while $\exists$ a fake edge in the solution
    Branch on endpoints of the fake edge
    Update C^*
  end if

end for

Return C^*

Fig. 15 Matching-based branch and bound algorithm for the maximum k-clique problem in UDG.

strictly greater than $k-1$. In fact, if the diameter of a connected component is less than or equal to k, that connected component already forms a k-clique, and there is no need for further computations.

In the main step, for each pair of adjacent vertices we form the subset of vertices $S = \{w \in V(G) : d(w,u) \leq d(u,v) \text{ and } d(w,v) \leq d(u,v)\}$. Since G_k is a UDG, the graph induced in G_k by the set S, $G_k[S]$, is a co-bipartite graph. Before solving the problem on this set, we check the degrees of vertices on the induced graph $G^k[S]$ and eliminate the ones that cannot take part in a solution better than the current best. If the size of the set S after vertex deletions is less than the current best, there is no need to solve the problem for this set and we continue with the next pair. Otherwise, we can identify the vertices that are neighbors to all the remaining vertices in $G^k[S]$. These vertices will be a part of the solution obtained from this set. We can solve the maximum clique problem for the remaining vertices in $G_k[S]$. This is done by forming the complement of the graph $G_k[S]$ and finding the maximum independent set in the complement by using matching. We will discuss the matching algorithm and how we obtain a maximum independent set in the implementation section. Now that we have the maximum clique in $G_k[S]$, we first check if the solution is better than the current best. Clearly, this solution is an upper bound for the maximum clique on $G^k[S]$. So, if it is not better than the current best, there is no need for further analysis and we move to the next pair. Otherwise, we check if this is a feasible solution for $G^k[S]$ by checking the existence of fake edges in the solution. If there exists a fake edge, this means that the endpoints of that edge are not adjacent on $G^k[S]$ and thus cannot be a part of the same clique. Thus, we create two branches for this problem by deleting an endpoint for each branch. We repeat the same procedure for each branch and thus have a branch-and-bound (B&B) tree. For each node, we fathom whenever we get a feasible solution for $G^k[S]$ or whenever we observe that it cannot give a solution

better than the current best. Thus, fathoming for the $B\&B$ tree for each pair of vertices is done with respect to both the global best and also the incumbent of the current $B\&B$ tree. Whenever we get a better solution, we update C^*. In the end, C^* corresponds to the maximum clique in $G^k[S]$, thus it is the maximum k-clique in G.

IV.4.1. Implementation

The algorithm is implemented in C++ using CPLEX callable library. The CPLEX defaults, such as cut generations, presolve and node heuristics, are all turned off. The branch and bound (B&B) framework of CPLEX requires an MIP formulation. The traditional B&B solves the LP relaxation of each node and updates the incumbent whenever it finds an integral solution better than the current incumbent. If an integral solution is found, the node is fathomed. Similarly, if the objective value of the LP relaxation is not better than the current best, the node is fathomed since the integral solution that it yields cannot be better than the LP relaxation. The branching is done on a variable which does not have an integral value. However, our problem has several different characteristics. First of all, we do not want to use k-clique formulation. Instead, we want to solve matching at each node, post-process the solution to obtain a maximum clique and decide on the feasibility by the presence of *fake* edges. Also, we want to branch on endpoints of *fake* edges. These are all possible by the flexibility provided by CPLEX callbacks.

In our initial implementation, we used the *incumbent callback* and the *branch callback*. CPLEX calls incumbent callback whenever it finds an integral solution better than the current best of the B&B tree. It enables the user to accept or reject the solution based on other user-defined factors. We used incumbent callback for checking for the presence of *fake* edges. The branch callback enables the user to create custom branches. We used this callback to create branches on endpoints of

fake edges. If there are not any fake edges or the solution is not promising for further analysis, the branch callback knows this by a flag from the incumbent callback and fathoms the node. If there are fake edges, it finds the vertex with the maximum number of fake edges and branches on that vertex and one of its *fake* neighbors.

Let $G' = \overline{G_k}[S]$ and $\ell(v)$ denote the set of edges for which the vertex v is an endpoint. The MIP formulation we used is the following:

$$\max \sum_{e \in E(G')} x_e + \sum_{v \in V(G')} x_v \tag{4.1}$$

subject to:

$$\sum_{e \in \ell(v)} x_e \leq 1 \; \forall v \in V(G') \tag{4.2}$$

$$x_v \in \{0,1\} \; \forall v \in V(G'), \qquad x_e \in \{0,1\} \; \forall e \in E(G') \tag{4.3}$$

All decision variables are binary. $x_e = 1$ if e is in the maximum matching and 0 if not. Without the decision variables corresponding to the vertices, this is a maximum matching formulation. In fact, there are not any constraints involving the variables x_v except for the binary restriction. Since this is a maximization problem, the optimal solution for all x_v's will be 1. The reason that these variables are introduced is to keep track of the deleted variables in set S for any node during the B&B process. This is helpful in post-processing the matching solution to obtain a maximum clique on the complement graph. The maximum matching is polynomial-time solvable. There are several algorithms studied in the literature for maximum matching in bipartite graphs. Another characteristic related to matching in bipartite graphs is that the polytope defined by the constraint matrix in the above formulation omitting the variables corresponding to vertices is totally unimodular. This means that the vertices of this polytope are all integral points and therefore an integral solution is guaranteed when

the LP relaxation is solved. We observe that introducing the x_v variables does not violate the totally unimodular property. Basically, including these variables in the formulation introduces an identity matrix to the constraint matrix. Let P be a totally unimodular matrix and I be an identity matrix. Then the matrix

$$\begin{bmatrix} P & 0 \\ 0 & I \end{bmatrix}$$

is also a totally unimodular matrix. Thus, by using the above formulation we can get an integral solution at each node of our $B\&B$ tree. This enables us not to worry about integrality constraints and focus on feasibility check by the presence of *fake* edges.

As this formulation involves many variables, we observed that the built-in simplex algorithm of CPLEX performs many iterations. Since there exist several efficient algorithms for maximum matching in bipartite graphs, we decided to include a matching algorithm in our implementation and avoid the above formulation. This implementation is possible by using the *solve callback*. Thus, our final implementation uses incumbent, branch and solve callbacks. The solve callback enables the user to identify a solution strategy, such as which CPLEX solver to use at a particular node. The solution obtained must be in CPLEX format. Thus, whichever algorithm we use, we need to make sure that CPLEX has some information about the solution status. Since our initial experiments showed improved performance on the latter implementation, we used it for further computational tests. However, in order for CPLEX to be able to create a B&B tree and perform branching, an MIP formulation is still needed. Therefore, we use the following formulation:

$$\max \sum_{v \in V(G')} x_v \tag{4.4}$$

subject to:

$$x_v \in \{0, 1\} \; \forall v \in V(G') \tag{4.5}$$

It is easy to see that an optimal solution for this formulation sets all variables to 1. The branching involves introducing an upper bound of 0 to the variables corresponding to the vertices that are the endpoints of a *fake* edge. At each node of the tree, we solve the above formulation with the corresponding bound changes using CPLEX LP solver. We give the solution to CPLEX since we know that the optimal solution sets all the variables to their upper bounds. Thus, CPLEX does not do any iterations and only confirms the optimality. Next, we solve the matching problem. Let $S' = \{w \in S : x^*_w = 1\}$, where x^*_w denotes the optimal value of the vertex w for the above formulation at the current node. The matching algorithm is run on the induced subgraph $G'[S']$. We use an algorithm based on augmenting paths. Given a *matching* M, an *alternating path* is a path in which the edges belong alternatively to the matching and not to the matching. An *augmenting path* is an alternating path that starts from and ends on unmatched vertices. One can prove that a matching is maximum if and only if it does not have any augmenting path. Let A' and B' denote the set of vertices corresponding to the partitions of $G'[S']$. Since a vertex in A' can only be matched with a vertex in B', it is sufficient to search through the unmatched vertices of A' to find an augmenting path. Figure 16 demonstrates how the solution is improved by augmenting paths. In (a), we find an augmenting path for a given matching M and by augmenting we increase the size of the matching in (b). The algorithm we used is outlined in Figure 17. We use breadth first search (BFS) for finding an augmenting path. As each path can be found in $O(|E|)$ time,

the running time of this algorithm is $O(|V||E|)$. We chose this implementation for its simplicity. Hopcroft-Karp algorithm for the same problem has a better running time of $O(\sqrt{|V|}|E|)$. Thus, the computational results may further improve by using Hopcroft-Karp algorithm.

Fig. 16 Matching by augmenting paths. (a) Matching M, red edges are in the matching and red vertices are matched vertices. There's an augmenting path. (b) Matching M' obtained by augmenting M. Now all vertices are matched.

Input: A bipartite graph G with partition sets A and B
Output: A maximum matching on G

$M = \{\}$
for $\forall w \in A$ **do**
  if w is unmatched
    Find an unmatched vertex $t \in B$ by BFS at vertex w
    if $\exists$ such t
    Augment the path from w to t and update M
    end if
  end if
end for
Return M

Fig. 17 Matching by augmenting paths on bipartite graphs.

Now that we have the maximum matching on $G'[S']$, we can find the maximum

independent set on $G'[S']$ (the maximum clique in $G_k[S']$). First we include all the unmatched vertices in the solution. It is clear that these vertices are all independent, since otherwise we would have an augmenting path and the matching we had would not be a maximum matching. For the matched vertices, we check their neighborhood and include them in the solution set if they do not have any neighbors in the current solution. In the end, we are guaranteed to have the maximum independent set.

For comparison purposes, we also implemented 1-plex formulation introduced in [10] in order to find the maximum clique of G^k. A subset of vertices S is said to be a k-plex if the following condition holds:

$$deg_{G[S]}(v) = |N(v) \bigcap S| \geq |S| - k \; \forall v \in S.$$

We chose this formulation because of its compactness compared to other formulations.

For a graph $G(V, E)$, let $\overline{d}_i = |V \setminus N[i]|$, where $N[i]$ denotes the set of neighbors of vertex i, including i itself. The binary variable $x_i = 1$ if i belongs to the maximum clique.

$$w(G) = max \sum_{i \in V} x_i \tag{4.6}$$

subject to:

$$\sum_{j \in V \setminus N[i]} x_j \leq \overline{d}_i(1 - x_i) \; \forall i \in V, \tag{4.7}$$

$$x_i \in \{0, 1\} \forall i \in V. \tag{4.8}$$

Table 1 Test instances for the maximum 2-clique problem.

input	no. of edges	density	no. of fake edges in G_2	initial soln	optimal soln
r100-1	4523	0.914	96	47	71
r100-2	4831	0.976	47	35	88
r100-3	4782	0.966	76	38	82
r100-4	4840	0.978	44	34	89
r100-5	4794	0.968	80	40	85
r100-6	1721	0.348	352	29	29
r100-7	2065	0.417	388	24	24
r100-8	2082	0.421	400	25	25
r100-9	2135	0.431	379	28	28
r100-10	2096	0.423	394	27	27
r200-1	18232	0.916	300	73	135
r200-2	18994	0.954	239	72	153
r200-3	19281	0.969	226	70	161
r200-4	19399	0.975	169	81	174
r200-5	19387	0.974	135	64	164
r200-6	7015	0.353	1182	41	41
r200-7	8249	0.415	771	49	50
r200-8	8713	0.438	828	53	55
r200-9	9032	0.454	807	60	62
r200-10	9487	0.477	770	55	56
r200-11	12754	0.641	1299	76	81
r200-12	14541	0.731	727	86	91
r200-13	15513	0.780	664	91	101
r200-14	15756	0.792	574	98	105
r200-15	15693	0.789	582	97	107
r500-1	116637	0.935	913	199	360
r500-2	119430	0.957	710	203	377
r500-3	121710	0.976	481	182	409
r500-4	122370	0.981	401	227	426
r500-5	121778	0.976	628	215	411
r500-6	49143	0.394	6158	124	124
r500-7	55019	0.441	2097	114	115
r500-8	59808	0.479	2662	118	120
r500-9	60373	0.484	3401	128	129
r500-10	58664	0.470	3249	125	125
r1000-1	79293	0.159	32724	96	96
r1000-2	112080	0.224	6260	106	106
r1000-3	123853	0.248	10936	113	116
r1000-4	129114	0.258	7799	119	119
r1000-5	129124	0.259	7674	113	114

IV.5. Computational Results

Both algorithms are implemented using C++ and CPLEX 11.0. The 1-plex formulation is solved by CPLEX with default settings. For both algorithms, an initial solution obtained by the greedy heuristic is used. Random test instances are generated and run on a Pentium 4 computer with 3.20 GHz CPU. We ran these algorithms for the maximum 2-clique problem. Table 1 displays the test instances. The optimal solution and the initial solution are also displayed on this table. $rx - y$ denotes the y^{th} test instance with x vertices. We created graphs with different edge densities. Densest cases have an edge density around 0.9 and sparsest cases - around 0.2.

The results are displayed in Tables 2 and 3. We observe that the 1-plex formulation outperforms our algorithm on dense instances. On the other hand, sparser instances favor the k-clique algorithm. For the 1-plex formulation we report the solution time (in seconds), number of B&B nodes and also the simplex iterations. Since the k-clique algorithm calls CPLEX for each pair that can lead to a better solution, we report the number of such calls as well as the maximum number of B&B nodes over all CPLEX calls, which gives an idea about memory usage. The CPLEX default settings are powerful in terms of reducing the B&B tree space by using several heuristics and cut generation. On the other hand, the structure of our k-clique algorithm does not allow us to do further processing at a single node. As a natural outcome, we search more nodes. But the results for instances $r500 - 6, \ldots, r500 - 10$ show that this is not always true. For the 1-plex formulation, CPLEX does many simplex iterations for these instances and in 4 out of 5 instances it has a larger B&B tree compared to the other algorithm. The average density of these instances is 0.45. The average solution time for the 1-plex formulation is 1921 seconds whereas it only takes 52 seconds on average for the k-clique algorithm.

Table 2 Comparison of 1-plex formulation and the k-clique algorithm. The solution time is in seconds.

	1-plex formulation			k-clique algorithm		
instance	sol. time	B&B nodes	simplex iter.	sol. time	max B&B nodes	CPLEX calls
r100-1	0.109	0	72	0.484	23	14
r100-2	0.094	0	13	0.031	0	0
r100-3	0.109	0	28	0.203	11	2
r100-4	0.078	0	13	0.125	0	1
r100-5	0.156	0	28	0.046	0	0
r100-6	0.546	0	410	0.016	0	0
r100-7	1.359	0	1524	0.047	0	0
r100-8	0.906	0	1114	0.031	0	0
r100-9	0.547	0	657	0.031	0	0
r100-10	0.547	0	654	0.031	0	0
r200-1	0.343	0	308	6.797	75	241
r200-2	0.141	0	168	3	7	97
r200-3	0.125	0	145	2.547	135	65
r200-4	0.109	0	50	1.157	3	3
r200-5	0.11	0	81	1.703	35	33
r200-6	58.249	53	15292	0.453	0	5
r200-7	32.281	14	7899	0.468	43	5
r200-8	14.906	5	3954	0.39	9	3
r200-9	4.406	0	2049	0.39	27	5
r200-10	15.953	0	4778	1.125	9	19
r200-11	1.484	0	762	2.156	323	5
r200-12	0.969	0	756	5.984	49	120
r200-13	0.609	0	618	4.657	41	110
r200-14	0.5	0	482	4.734	15	112
r200-15	0.719	0	744	5.562	67	205
r500-1	1.141	0	674	188.966	421	1655
r500-2	0.64	0	714	177.779	641	1865
r500-3	0.344	0	356	102.17	41	799
r500-4	0.312	0	354	70.031	3323	429
r500-5	0.281	0	385	102.045	3175	965
r500-6	3086.9	64	53000	37.656	1023	356
r500-7	2109.69	475	121426	35.452	247	467
r500-8	1497.96	554	124515	98.218	157	2050
r500-9	1288.05	475	113188	49.187	41	428
r500-10	1622.36	475	209336	40.609	63	448

Table 3 Performance of 1-plex formulation on random UDGs with 1000 vertices. The solution time is in seconds.

instance	k-clique alg. sol. time	time limit	best soln	soln gap (%)
r1000-1	6.218	3978.46	96	80.45
r1000-2	66.593	3181.98	106	78.81
r1000-3	112.092	2952.22	113	76.81
r1000-4	120.811	2838.83	119	75.54
r1000-5	182.935	2872.34	113	76.76

For larger instances, we focus on sparser cases. In fact, if we consider wireless network applications, it is more likely to have sparser instances since denser instances will yield more signal interference and also require more energy. When we run our algorithms for graphs with 1000 vertices with an edge density around 0.2, we observe that 1-plex formulation fails to give the optimal solution in a reasonable time. So we introduce time limits for CPLEX. On the other hand, we can easily get the optimal solution by the k-clique algorithm. Table 3 displays the solution time for the k-clique algorithm to find the optimal solution as well as the results that 1-plex formulation achieves within the time limit set. On the average, our algorithm finds the optimal solution in 98 seconds whereas on an average of 3164 seconds, the 1-plex formulation only shows that the solution it provides has a gap which is more than 75%.

During our implementation of the k-clique algorithm, we also observed that whenever we do a preprocessing on the set S by deleting the vertices with a small degree on the graph induced by set S, the performance of the algorithm improves significantly. As displayed in Table 4, both the number of CPLEX calls and the running time decrease. Figure 18 shows the improvement in the running time. The curve labeled "with" shows the data obtained by doing the preprocessing. The x-axis corresponds to the test instances in the order presented in Table 4.

Table 4 Comparison of the k-clique algorithm with and without preprocessing set S. The solution time is in seconds.

	without preprocessing			with preprocessing		
instance	sol. time	B&B nodes	CPLEX calls	sol. time	B&B nodes	CPLEX calls
r100-1	1.75	15	182	0.484	23	14
r100-2	0.266	7	20	0.031	0	0
r100-3	0.828	69	90	0.203	11	2
r100-4	0.281	5	22	0.125	0	1
r100-5	0.359	79	28	0.046	0	0
r100-6	0.156	3	9	0.016	0	0
r100-7	0.641	63	59	0.047	0	0
r100-8	0.391	11	38	0.031	0	0
r100-9	0.188	3	14	0.031	0	0
r100-10	0.266	3	20	0.031	0	0
r200-1	15.234	85	862	6.797	75	241
r200-2	7.953	29	569	3	7	97
r200-3	6.032	139	439	2.547	135	65
r200-4	2.61	13	174	1.157	3	3
r200-5	5.015	61	395	1.703	35	33
r200-6	2.562	15	252	0.453	0	5
r200-7	1.422	9	142	0.468	43	5
r200-8	1.234	13	121	0.39	9	3
r200-9	1.031	45	88	0.39	27	5
r200-10	4.969	9	439	1.125	9	19
r200-11	8.86	33	1 514	2.156	323	5
r200-12	17.672	39	891	5.984	49	120
r200-13	16.219	41	886	4.657	41	110
r200-14	12.89	11	769	4.734	15	112
r200-15	15.845	117	885	5.562	67	205
r500-1	334.762	335	4793	188.966	421	1655
r500-2	305.621	1719	4810	177.779	641	1865
r500-3	164.03	25	3249	102.17	41	799
r500-4	122.749	3323	2446	70.031	3323	429
r500-5	185.247	2641	3172	102.045	3175	965
r500-6	94.718	1025	2091	37.656	1023	356
r500-7	192.185	247	4257	35.452	247	467
r500-8	399.057	117	6905	98.218	157	2050
r500-9	308.153	9	4732	49.187	41	428
r500-10	163.248	131	4100	40.609	63	448

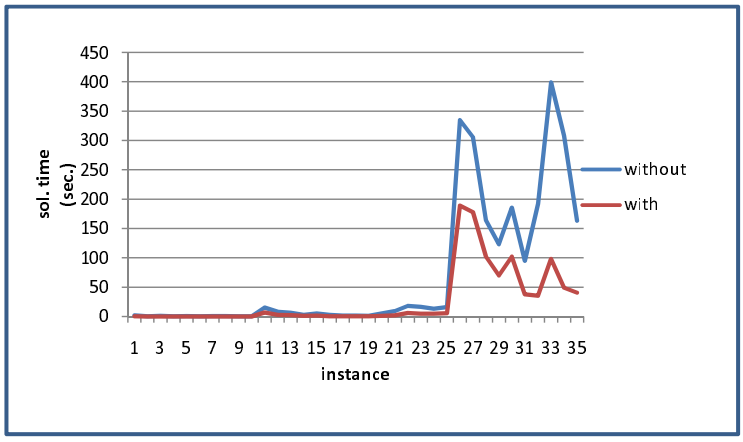

Fig. 18 The effect of preprocessing the set S (deleting vertices of small degree) on running time of the k-clique algorithm.

IV.6. Conclusion

In this chapter, we studied the maximum k-clique problem in unit disk graphs. We conjecture that this problem is NP-hard. To support this conjecture, we presented some complexity results of the clique problem in geometric graphs that are closely related to the k-clique problem in unit disk graphs. We propose an exact solution procedure, which is a matching-based branch and bound method. We report the results of computational experiments on randomly generated test instances. For comparison purposes, we also solved this problem with CPLEX defaults by using the 1-plex formulation on the k^{th} power of the input graph. The test results show the effectiveness of our algorithm, especially for the sparser instances.

As a future work, we would like to prove the computational complexity. We are also interested in designing efficient approximation schemes. Analyzing this problem in unit ball graphs is another task for future work. We believe that with some

modifications to our exact solution procedure, we can also obtain a solution procedure for the maximum k-club problem.

CHAPTER V

THE MINIMUM K-BCDS PROBLEM IN UDGs

Typically, a wireless network is modeled as a *unit-disk* (*unit-ball*) graph, in which vertices are given by points on the plane (in 3-dimensional Euclidean space), and two vertices are connected by an edge if and only if the corresponding pair of points has at most one-unit distance between them. Here the unit of distance represents the transmission range of a wireless node, which is assumed to be the same for each node. There are several variations of these popular geometric graphs that are used to model wireless network topology, such as double-disk graphs, quasi-unit disk graphs, unit ball graphs in a doubling metric and bounded independence graphs. However, one common feature of the traditional models is the assumption that the range of communication or its estimate is given in advance and is used as an input. Since for some types of wireless networks, such as sensor networks, energy considerations are extremely critical, this assumption becomes especially important.

In a wireless network, each node typically has transmission and reception processing capabilities. The transmission of a signal between two nodes requires power for the following [63]:

- The source node needs power to prepare the signal.
- Power is needed to support the link between two nodes. Let u and v be wireless nodes with Euclidian distance $d_{u,v}$ in between. The power needed is proportional to $d_{u,v}^{\beta}$, where β is a real constant between 2 and 5 dependent on the transmission environment. This is also called a *path loss*.
- The receiving node needs some power to receive, store and then process that

signal.

Path loss is the step that requires the most power among others. Suppose we have three nodes a, b and c such that $d_{a,b} = d_{b,c} = 1$ and $d_{a,c} = 2$. If we consider the path loss formula, we observe that the transmission between nodes a and c can be more power efficient if an indirect path a-b-c is chosen instead of a direct transmission path a-c. Thus, determining the transmission range that minimizes the energy used is an important decision problem.

The network topology, which is crucial in designing routing protocols, is completely defined by the transmission range. Therefore, the choice of the transmission range has to be approached with utmost care. Several papers investigate this problem with the goal of ensuring connectivity. Ideally, one would want to tie the choice of transmission range to a specific routing protocol that will be used for communication within the network. One of the most popular approaches in designing routing protocols is connected to the concept of *virtual backbone*, which is a (small) subset of nodes that are used as a core for communication within the network. In particular, connected dominating sets are often used to describe a virtual backbone in ad hoc wireless networks.

We propose to use the bottleneck connected dominating set problem as a viable approach to selecting the transmission range of a wireless node in a network.

Given a complete graph $G = (V, E)$ with positive edge weights (costs, distances) $c_e, e \in E$, the bottleneck subgraph $G(e)$ of G corresponding to the edge e is defined as follows:

$$G(e) = (V, E(e)), \text{ where } E(e) = \{e' \in E : c_{e'} \leq c_e\}.$$

Then the cost of the bottleneck subgraph $G(e)$ is defined as the cost of the maximum weight edge in $E(e)$, i.e., $cost(G(e)) = c_e$. Given a subset of vertices S in G, its

bottleneck connected domination cost, $cost(S)$, is defined as the minimum cost of a bottleneck subgraph $G(e)$ of G such that S forms a connected dominating set in $G(e)$. The k-bottleneck connected dominating set (k-BCDS) problem is to find a subset of k vertices with minimum bottleneck connected domination cost in G. A restricted version of k-BCDS problem, in which the edge weights are required to satisfy the triangle inequality, will be denoted by k-BCDS(Δ).

V.1. Literature Review

The problem of designing an energy-efficient wireless network has been studied by many researchers. Most of the papers in this research area focus on the packet radio network model of wireless networks. The main difference from the unit disk model is that transmissions of messages *interfere* at a node if at least two of its neighbors transmit a message at the same time. The analysis of different designs are mostly tested by simulation experiments.

Deng et al. [30] consider the radio transmission range as a static system parameter determined a priori, i.e., during system design, and used throughout the lifetime of a wireless ad hoc network. Assuming uniformly distributed network nodes, they show that the optimum transmission range is influenced more by node density than the network coverage area. The path-loss exponent is also an important parameter in determining the optimal range. The authors observe that when the path-loss exponent is four, the optimal transmission ranges are almost identical over different node densities in their experiments. When the exponent is two, they observe that the optimal transmission range decreases as the node density increases.

Wu et al. [82] focus on the connected dominating set to optimize energy consumption via transmission range reduction. The main motivation for their approach

is that in general, nodes in the CDS consume more energy to handle various bypass traffic than nodes outside the set. Thus, to prolong the life span of each node and, hence, the network, nodes should be alternated, when possible, to form a CDS. The objective of this research is to devise a selection scheme (for dominating hosts) so that the overall energy consumption is balanced in the network, and at the same time, a relatively small connected dominating set is generated. The authors use a simple localized CDS algorithm which is based on the so called *marking process* in which a node is included in the CDS if two of its neighbors are not directly connected. This method is proposed by Wu and Li [81]. After this marking process some rules are applied to remove a vertex from the CDS, thus decreasing the size of the CDS. In [82] the authors consider some extended rules for reducing the CDS size by removing certain nodes. The authors assume that initially the network is connected under the uniform transmission range. Each node dynamically reduces its transmission range when possible based on neighbor distance information and the neighbor set of each neighbor or just the forwarding neighbor, which is the closest neighbor. Next, they update the CDS based on the above technique by using node elimination rules. They run simulation experiments and compare the effect of different node elimination rules on the CDS size and the energy consumption of the network.

Rodoplu and Meng [69] propose a local optimization scheme that finds the minimum energy links and dynamically updates them. They consider a directed graph, and the goal is guaranteeing strong connectivity and minimum energy consumption in the network. A directed graph is strongly connected if there exists a path between any pair of vertices in the graph. They dynamically compute a sparse and strongly connected graph of communication links between all the nodes by using only local information, and the existing links will be only between nodes that are close enough to be neighbors. Their scheme is both applicable for stationary and mobile networks.

The efficiency of their scheme is supported by simulation experiments.

Li and Halpern [62] propose an improvement to the method proposed by Rodoplu and Meng [69] by relaxing the assumption in the former one, which states that the transmission region is a circular region.

Sanchez et al. [71] present an algorithm to calculate the minimum transmission range of the transceivers that is required to achieve, with some probability, full network connectivity. They identify the critical link for which the removal will partition the graph. Given two network nodes a and b, the authors describe a and b as direct neighbors if there is no other node c that is closer to a than b is and vice versa. They find the critical range by considering the direct neighbor graph. For each loop of this graph, they remove the maximum weight link. Then the maximum weight of the remaining edges corresponds to the critical range. Although the authors do not state this explicitly, this problem is a minimum bottleneck spanning tree problem.

The k-BCDS problem that we propose optimizes the transmission range by ensuring a predetermined connected dominating set of size k. To our best knowledge, this problem has not been studied before in the context of wireless networks. We assume that transmission ranges of all nodes are the same and a unit disk graph model is used.

In the context of graph theory, there are several bottleneck problems studied, such as minimum bottleneck spanning tree, bottleneck traveling salesman problem, etc. The bottleneck version of the dominating set problems that was studied previously focuses on vertex weighted graphs, where bottleneck cost is defined in terms of the vertex weights. Yen [84] introduces the bottleneck dominating set and bottleneck independent dominating set problems, where the bottleneck is defined as the maximum weighted vertex. Thus, the goal is finding a dominating set in which the maximum weight of any vertex in the dominating set is minimized. The author shows

that this problem can be solved in $O(n \log n + m)$ time. On the other hand, the bottleneck independent dominating set problem is proven to be NP-hard on planar graphs. Kloks et al. [59] present linear-time algorithms for minimum bottleneck dominating set and minimum bottleneck total dominating set problems. They provide polynomial algorithms for the minimum bottleneck independent dominating set problem restricted to chordal graphs, split graphs, permutation graphs and graphs of bounded treewidth. They also state that the minimum bottleneck connected dominating set problem can be solved in $O(m \log n)$ time.

In our problem, we define the bottleneck cost in terms of the edge weights. To our best knowledge, there are no published results on the edge-weighted bottleneck dominating set problems.

V.2. Complexity and Approximation

We show that the approximation of the k-BCDS(Δ) problem with a factor $2 - \epsilon$ is NP-hard. This also shows that the k-BCDS problem is NP-hard on general graphs.

Proposition 1. *It is NP-hard to approximate the k-BCDS(Δ) problem within a factor $2 - \epsilon$ for any $\epsilon > 0$.*

Proof. The reduction is from the minimum connected dominating set problem in general graphs. Given a graph G, we form a new edge-weighted complete graph G' such that $V(G) = V(G')$ and the weight of an edge e, w_e, is 1 if $e \in E(G)$ and 2 otherwise. Clearly, the edge weights of G' satisfy the triangle inequality. Now suppose that we have a $2 - \epsilon$ approximation algorithm, A, for the k-BCDS(Δ) problem. If we apply this algorithm to G', we have the following possibilities. Case 1: $A(G') = 2$. This means that the bottleneck cost of 2 is optimal, since if it was not then the optimal objective value would be 1, so $A(G') = 2$ would be impossible. This leads us

to the conclusion that the cardinality of a minimum CDS in G is strictly greater than k. Case 2: $A(G') = 1$. This means the optimal solution for minimum k-BCDS(Δ) in G is 1. Thus there exists a CDS in G of size less than or equal to k.

Considering these two cases, the minimum connected dominating set problem can be solved in polynomial time for any graph G by applying the algorithm A. Thus, it is NP-hard to approximate the k-BCDS(Δ) problem within a factor $2 - \epsilon$ for any $\epsilon > 0$, unless $P = NP$. □

Proposition 2. *There exists a 3-approximation algorithm for the k-BCDS(Δ) problem.*

In order to prove this proposition, we present an approximation algorithm and show that its performance ratio is at most 3. The proposed algorithm is similar to the technique developed by Hochbaum and Shmoys for bottleneck problems [52] and is outlined in Figure 19. The first step of the algorithm is to sort the edge weights. Next, we solve the minimum bottleneck spanning tree problem in order to find the minimum edge weight that guarantees connectivity in the graph. This is a lower bound for the k-BCDS. We can further improve this bound in Step 2, by the following argument:

Lemma 14. *Let c_{e^*} be the optimal bottleneck cost for the k-BCDS(Δ) on a graph G and let the edge (u, v) have the largest cost, $c(u, v)$, in G. Then we have:*

$$c_{e^*} \geq c(u, v)/(k + 1).$$

Proof. Since in the optimal solution we will have a CDS of size k, the shortest path between any pair of vertices can have at most k internal vertices in the corresponding bottleneck graph $G(e^*)$. Since edge weights in G satisfy the triangle inequality, we have:

$$c(a, b) \leq c(P^*_{a,b}) \leq (k + 1)c_{e^*} \forall (a, b) \in E(G),$$

Input: $G = (V, E)$ with edge weights satisfying the triangle inequality; k
Output: A k-BCDS in G

0. Sort all edges in E in nondecreasing order of their costs:

$$c_{e_1} \leq c_{e_2} \leq \cdots \leq c_{e_m}, m = \binom{|V|}{2}.$$

1. Compute a minimum bottleneck spanning tree T of G with $cost(T) = c_{\tilde{e}}$;
2. **If** $c_{\tilde{e}} < c_{e_m}/(k+1)$ **then** set $c_{\tilde{e}} = c_{e_m}/(k+1)$;
3. Let e_i be the edge weight such that $c_{e_i} - c_{\tilde{e}}$ is nonnegative and minimal;
4. Set $\hat{G} = G(e_i)$;
5. Compute a maximal independent set I of $\hat{G}^2$ as follows
   $I = \{v\}$ where v is an arbitrary vertex in $\hat{G}$
   while I is not maximal
   Choose a feasible vertex w in $\hat{G}$ such that $\exists u \in I$: $d_{\hat{G}}(u, w) = 3$
   $I = I \bigcup \{w\}$
   end while
6. **If** $|I| > k$
   $i = i + 1$;
   go to step 4;
end if
7. Return I.

Fig. 19 The 3-approximation algorithm for k-BCDS(Δ).

where $c(a, b)$ is the cost of the edge (a, b) and $c(P^*_{a,b})$ is the total cost of the shortest path from the vertex a to the vertex b on the bottleneck graph $G(e^*)$. Therefore we have:

$$c_{e^*} \geq c(u, v)/(k+1).$$

□

In Step 4 of our approximation algorithm, we form the bottleneck graph corresponding to the edge e_i, $\hat{G} = G(e_i)$. We compute a maximal independent set I on $\hat{G}^2$ by starting from an arbitrary vertex. We expand the set I by including a vertex such

that it is in the 3-hop neighborhood of some vertex of I in $\hat{G}$ and not a neighbor of any vertex of I in $\hat{G}^2$. We stop when it is not possible to expand I, i.e., I is a maximal independent set in $\hat{G}^2$. If $|I| \leq k$, we can claim that I is a connected dominating set of size less than or equal to k on the bottleneck graph $G(e')$, where $c_{e'} = 3c_{e_i}$. Otherwise, we pick the next edge weight in G and continue our search.

Proposition 3. *For any given instance of the k-BCDS(Δ) problem we have*

$$cost(I) \leq 3cost(D^*),$$

where D^* denotes an optimal solution of k-BCDS(Δ) problem for this instance.

Proof. By the construction of the set I, it is easy to see that each time we add a vertex to set I, it is at most at distance $3c_{e_i}$ to some vertex in I. Thus, the output set I is of size $\leq k$ and is a connected dominating set in $G(e')$, where $c_{e'} = 3c_{e_i}$. It is easy to see that $cost(D^*) \leq cost(I)$ since I is a feasible solution and D^* is an optimal solution. Furthermore, if $c_{e_i} \leq cost(I)/3$, then the corresponding bottleneck graph $G(e_i)$ has a maximal independent set of cardinality greater than k. This indicates that the corresponding bottleneck graph cannot have a connected dominating set of size $\leq k$. Thus, we have:

$$\begin{aligned} cost(D^*) &\leq cost(I), \\ cost(D^*) &\geq cost(I)/3. \end{aligned}$$

□

Minimum bottleneck spanning tree: The 3-approximation algorithm for the k-BCDS(Δ) problem requires the computation of a minimum bottleneck spanning tree. A spanning tree T of G is a minimum-bottleneck spanning tree if there is no

spanning tree T of G with a cheaper bottleneck edge.

Lemma 15. *Given a graph G, any minimum spanning tree of G is also a minimum bottleneck spanning tree.*

Proof. Given a graph G, let T denote the minimum spanning tree in G. Assume that there exists a spanning tree T' which has a cheaper bottleneck edge. Let the edge (a, b) denote the bottleneck edge of T. Consider the trees T_a and T_b obtained from T by the removal of the edge (a, b). Since T' is also a spanning tree of G, there exists an edge in T' that connects T_a and T_b. Furthermore, the cost of this edge is less than or equal to the bottleneck cost of T' and hence it is less than $c(a, b)$. Hence, there exists a spanning tree with cost less than the cost of T. This is a contradiction with T being the minimum spanning tree. Thus, every minimum spanning tree is also a minimum bottleneck spanning tree. □

Based on Lemma 15, we can use a minimum spanning tree algorithm such as Prim's algorithm or Kruskal's algorithm to find the minimum bottleneck spanning tree.

Search procedure: Given a graph G, it is easy to see that as we increase the bottleneck cost, the size of a minimum connected dominating set is non-increasing. Our goal is to find the minimum bottleneck cost at which we have a connected dominating set of size k. Thus, instead of incrementing the bottleneck cost one by one in the order of edge costs, we can use a better search technique such as the binary search method.

V.2.1. Complexity in UDGs and UBGs

Theorem 17. *The minimum k-BCDS can be solved in polynomial time in unit disk and unit ball graphs for any fixed $k > 0$.*

Proof. Let G be a unit disk or a unit ball graph and k a given constant. The edge costs are the pairwise distances between vertices. It is easy to see that the edge costs satisfy the triangle inequality. Thus, the following steps can give us the optimal solution:

- Use the 3-approximation algorithm to determine the range $[e', 3e']$ efficiently.
- Apply a search procedure in the interval $[e', 3e']$ by using the PTAS for the minimum connected dominating set problem [85] in UBGs by setting:

$$\epsilon = 1/(k+1).$$

Let S_e be the connected dominating set found by the PTAS for the bottleneck graph corresponding to cost e, and let O_e be the minimum connected dominating set for the same problem. It is easy to see that:

$$|S_e| > k \quad \Leftrightarrow \quad |O_e| > k,$$
$$|S_e| \leq k \quad \Leftrightarrow \quad |O_e| \leq k.$$

Since $\epsilon = 1/(k+1)$ and $|O_e| \geq |S_e|/(1+\epsilon)$, whenever $|S_e| > k$ (which means $|S_e| \geq k+1$) we have:

$$|O_e| \geq |S_e|/(1+\epsilon) = |S_e|\frac{k+1}{k+2},$$
$$|O_e| \geq \frac{(k+1)^2}{k+2} = k + \frac{1}{k+2}.$$

Thus, we terminate the search at the smallest edge cost in $[e', 3e']$ for which

$S_e \leq k$. The running time of each step is $O(n^{(k+1)^3})$ since the PTAS runs in $O(n^{(k+1)^3})$ time and it is the most time-consuming step. We need to run the PTAS at most for n^2 times. Therefore, although this is not a practical solution procedure, for a constant k the running time is polynomial.

□

V.3. Conclusion

In this chapter, we studied the bottleneck connected dominating set problem. Motivated by the wireless network applications, we propose this problem as a viable approach to determine an optimal transmission range for a network. We presented the work done in the literature in terms of optimizing the transmission range with respect to several other factors such as connectivity. We observed that the previous approaches did not focus on the problem when a predetermined size of a "virtual backbone" is sought. A "virtual backbone" corresponds to a connected dominating set in a graph-theoretic representation of the network. The minimum k-BCDS problem seeks a minimum edge weight in the graph such that the corresponding bottleneck graph has a connected dominating set of size k. We also observed that this problem has not been studied in the graph theory literature. The vertex weighted version is known to be polynomial-time solvable. We proved that this problem is NP-hard even when it is restricted to graphs whose edge weights satisfy the triangle inequality. Furthermore, we also showed that it is not approximable within a factor of $2 - \epsilon$ even for graphs whose edge weights satisfy the triangle inequality. We proposed a 3-approximation algorithm for graphs that satisfy the triangle inequality. Finally, we proved that this problem is solvable in $O(n^{2+(k+1)^3})$ when the input graph is a unit disk or a unit ball graph. Our future goal is to find more efficient solution procedures for unit ball graphs.

CHAPTER VI

HEURISTIC JUSTIFICATION

VI.1. Introduction

Due to the inherent computational complexity of most combinatorial optimization problems, one cannot hope to solve very large-scale instances to optimality and (meta) heuristics are usually applied in practice. In many cases, researchers and practitioners rely on variations of greedy heuristics that are very simple to understand and implement. This simplicity and effectiveness of heuristic approaches earned them a considerable popularity in optimization community. On the other hand, there is a fair amount of skepticism towards such approaches due to a lack of theoretical foundations behind them. Indeed, in most cases there is no provable approximation ratio for the performance of heuristics, meaning that the computed solution may potentially be arbitrarily far away from the optimum. Then a reasonable question to ask is, what is the reason a particular simple heuristic is chosen as the solution approach for a given problem? Is this heuristic really the "best" choice? For some problems that allow constant-ratio approximation algorithms, but are hard to approximate within a given constant factor, questions of this type have been answered as follows. Assume that we know that a problem $\mathcal{P}$ is hard to approximate within a factor better than c, where c is a given constant. Then any polynomial-time algorithm $\mathcal{A}$ that approximates $\mathcal{P}$ within the factor of c can be claimed to be the "best" in the sense that no other polynomial-time algorithm can be provably better than $\mathcal{A}$, unless $P = NP$. However, there are many problems associated with the hardness of approximation results claiming that these problems are hard to approximate within any constant

factor. In this case, we cannot claim that a certain heuristic is the "best" based on an approximation ratio. We propose an approach to justify the usage of certain greedy heuristics in such cases, which extends the common definition of the "best approximation algorithm" to the problems for which a constant-ratio approximation algorithm is unlikely to be found. Namely, given a problem $\mathcal{P}$ and a heuristic algorithm $\mathcal{A}$ for this problem, we can claim that this heuristic is the "best" for the given problem if finding a solution better than that output by $\mathcal{A}$ (whenever one exists) is NP-hard. To prove such a result it suffices to show that it is NP-hard to recognize whether there is a gap between the optimal objective function value and the value of the solution output by $\mathcal{A}$. Our research is motivated by the following result.

Let $\bar{\omega}_k(G)$ denotes the k-club number of G. Clearly, for $l < k$ we have

$$\bar{\omega}_l(G) \leq \bar{\omega}_k(G).$$

For a simple undirected graph and a given positive integer k, a k-club is a subset of vertices that induces a subgraph of diameter at most k, and the k-club number $\bar{\omega}_k(G)$ is the cardinality of a largest k-club in G. Note that for $l = 1$ an l-club is a clique and $\bar{\omega}_1(G) \equiv \omega(G)$, where $\omega(G)$ is the clique number of G. In this case, for $k \geq 2$, we have an obvious inequality:

$$\omega(G) \leq \Delta(G) + 1 \leq \bar{\omega}_k(G),$$

in which, similarly to the famous Sandwich Theorem, a polynomially-computable value is sandwiched between two values that are NP-hard to compute. We have proved the following statement:

Proposition 4. *Let positive integers k and l, $l < k$ be given. The problem of checking whether $\bar{\omega}_l(G) = \bar{\omega}_k(G)$ is NP-hard.*

Using this proposition, we obtain the following corollary.

Corollary 2. *Let k be a fixed integer, $k \geq 2$. Unless $P = NP$, there cannot be a polynomial time algorithm that finds a k-club of size greater than $\Delta(G)+1$ whenever such a k-club exists in the graph.*

Corollary 2 can be used as a reasonable theoretical justification of using a simple heuristic (based on finding a maximum-degree vertex) for the maximum k-club problem with fixed $k \geq 2$. It may also be viewed as an additional evidence of the problem's practical intractability. On the other hand, this should not prevent the practitioners from designing more sophisticated approaches for the maximum k-club problem, since the above result describes just the worst-case behavior of the heuristics. Simple greedy heuristics are often used to solve large-scale instances of NP-hard problems in practice, and similar complexity results for other problems would be a good way to explain the choice of the approach from theoretical perspective.

In the remainder of this chapter, we present the proof of Proposition 4 and show how we obtain Corollary 2. Similar results will be presented for the maximum clique, maximum independent set, maximum k-plex and minimum vertex coloring problems. We also investigate the more general node deletion problems (maximum induced subgraph with some property P) with respect to provably best heuristics.

VI.2. The Maximum k-Club Problem

Proof of Proposition 4: Assume that there is a polynomial time algorithm $\mathcal{A}_{kl}(G)$ that, given a graph G , correctly answers the question "Is $\bar{\omega}_l(G) = \bar{\omega}_k(G)$?" with either "yes" or "no". Next we analyze the two possible cases.

(i) The answer given by $\mathcal{A}_{kl}(G)$ is "no". Then we run the following polynomial-time algorithm to compute $\bar{\omega}_k(G)$:

0. $G' = G,\ i = 1$;

1. while the answer of $\mathcal{A}_{kl}(G)$ is "no" repeat

   $G' = G' \cup C_i$, where C_i is the clique on i vertices;

   $i = i + 1$.

2. return i.

Note that starting from $i = \bar{\omega}_l(G)$ each next iteration of this algorithm will increase $\bar{\omega}_l(G')$ by 1 without changing $\bar{\omega}_k(G') = \bar{\omega}_k(G)$. The algorithm will terminate when $i = \bar{\omega}_l(G') = \bar{\omega}_k(G') = \bar{\omega}_k(G)$.

(ii) The answer given by $\mathcal{A}_{kl}(G)$ is "yes". Again, we want to design a polynomial time algorithm for computing $\bar{\omega}_k(G)$. Since k is a constant, we can check if $\bar{\omega}_k(G) < k$ by examining all subsets of size $< k$ in $O(n^k)$ time. Thus, we are interested only in the case where $\bar{\omega}_k(G) \geq k$. Denote by B_s^h the (h, s)-*broom graph*, which consists of a path of h vertices and s more vertices connected to one of the endpoints of this path. As an example, Figure 20 shows the broom graph B_4^3. Obviously, the diameter of B_s^k is equal to k if $k \geq 1$, therefore, for any $l < k$, B_s^k is a k-club, but not an l-club. We can use the broom graphs to compute $\bar{\omega}_k(G)$ as follows.

0. $G' = G,\ i = 1$;

1. while the answer of $\mathcal{A}_{kl}(G)$ is "yes" repeat

   $G' = G' \cup B_i^k$;

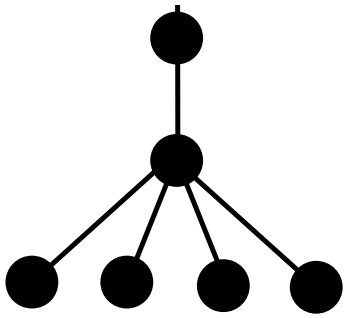

Fig. 20 A sample broom graph B_4^3.

$i = i + 1$.

2. return $i + k - 1$.

Hence, we have shown that if $\mathcal{A}_{kl}(G)$ was a polynomial-time algorithm, then we would be able to compute $\bar{\omega}_k(G)$ in polynomial time. The result follows from NP-hardness of computing $\bar{\omega}_k(G)$ for any fixed k. □

Next we discuss the implications of the above complexity result for the case when $l = 1$ and $k \geq 2$. Note that $\omega(G) \leq \Delta(G) + 1 \leq \bar{\omega}_k(G)$ and we can easily check whether $\omega(G) = \Delta(G) + 1$ by checking, for each maximum degree vertex in G, if its neighbors form a clique. Hence, it is NP-hard to check whether $\bar{\omega}_k(G) = \Delta(G) + 1$. This implies that, unless $P = NP$, one cannot design a polynomial-time heuristic for the maximum k-club problem, which is provably better than the trivial approach consisting in picking a maximum degree vertex and all its neighbors as the output k-club. Indeed, existence of a polynomial-time algorithm that finds a k-club of size greater than $\Delta(G) + 1$ whenever $\bar{\omega}_k(G) > \Delta(G) + 1$ would imply that one can check in polynomial time whether $\bar{\omega}_k(G) = \Delta(G) + 1$. Thus, we obtain Corollary 2.

Assume that one needs to solve the maximum k-club problem in G for some fixed large k. If one is given a polynomially computable parameter $v_{l,k}(G)$ such that

$\bar{\omega}_l(G) \leq v_{l,k}(G) \leq \bar{\omega}_k(G)$, it seems natural to expect a higher value of $v_{l,k}(G)$ to correspond to a higher the value of l. However, since for $l \geq 2$ we have

$$\omega(G) \leq \Delta(G) + 1 \leq \bar{\omega}_l(G) \leq v_{l,k}(G) \leq \bar{\omega}_k(G),$$

Corollary 2 implies that we cannot expect that $v_{l,k}$ will dominate $\Delta(G)+1$, no matter how high the value of $l < k$ is.

In a special case when $k = 2l$, we can use $v_{l,k}(G) = \Delta(G^l) + 1$, where G^l is the l^{th} power of graph G, which is defined on the same set of vertices as G with edges connecting all pairs of vertices that are distance at most p from each other. It is easy to see that $\bar{\omega}_l \leq \Delta(G^l) + 1 \leq \bar{\omega}_{2l}$ and $\bar{\omega}_l = \Delta(G^l) + 1$ if and only if the neighbors of one of the vertices of degree $\Delta(G^l) + 1$ in G^l form a clique in G^l (which can be easily checked). Therefore, checking if $\Delta(G^l) + 1 = \bar{\omega}_{2l}$ is NP-hard. Note that for two positive integers p and r, such that $p > r > 1$, we have $\Delta(G) \leq \Delta(G^r) \leq \Delta(G^p)$ and

$$\omega(G) \leq \Delta(G) + 1 \leq \bar{\omega}_r(G) \leq \Delta(G^r) + 1 \leq \Delta(G^p) + 1 \leq \bar{\omega}_{2p}(G).$$

Since $\Delta(G^r) < \Delta(G^p)$ implies $\bar{\omega}_r(G) < \bar{\omega}_{2p}(G)$, the problem of checking whether $\bar{\omega}_r(G) = \bar{\omega}_{2p}(G)$ remains NP-hard even if restricted to graphs with $\Delta(G^r) = \Delta(G^p)$.

VI.3. Maximum Independent Set and Maximum Clique Problems

It is not possible to approximate the maximum independent set problem within a factor of Δ^ϵ for some $\epsilon > 0$ [6]. Similarly, the maximum clique problem is not approximable within a factor of $n^{1-\epsilon}$ [49]. In an interesting related paper [21], Busygin and Pasechnik have shown that it is NP-hard to check whether $\bar{\chi}(G) - \alpha(G) = 0$, where $\bar{\chi}(G)$ denotes the cardinality of a minimum clique partitioning in G and $\alpha(G)$ is the independence number of G. This is in contrast to the fact that checking whether a graph

G is perfect (i.e., for every subset S of vertices we have $\bar{\chi}(G[S]) = \alpha(G[S])$) is polynomial [28]. The Busygin-Pasechnik result implies that any polynomially-computable parameter that lies between $\alpha(G)$ and $\bar{\chi}(G)$ will provide the "best" upper bound for the independence number in the sense that no other polynomially computable bound can be provably better for all graphs where this bound can be improved. In particular, the Lovász theta is one such polynomially computable bound [42].

Since a maximum independent set in a graph G corresponds to the maximum clique problem in $\overline{G}$ and a minimum clique partitioning in G corresponds to a minimum vertex coloring in $\bar{(}G)$, the above complexity result also applies to the maximum clique and minimum vertex coloring problems. This result is in terms of a provably "best"upper bound for the maximum clique and maximum independent set problems. Now, we outline a sequential greedy heuristic for the maximum clique problem and prove that we cannot have an approximation algorithm provably better than our greedy heuristic, unless $P = NP$.

The greedy heuristic outlined in Figure 21 first picks a maximum degree vertex and includes it in the current clique C. The remaining steps are repeated till the maximality is achieved. The set S denotes the set of possible candidates for inclusion in the clique. Each vertex in S is a neighbor for all the vertices in the current clique C. Among these candidates, we choose the vertex with maximum degree on the induced graph $G[S]$ and include it in the solution. We repeat this procedure until the set S is empty. When S is empty, we attain maximality since there does not exist any vertex that can be included in the solution.

Proposition 5. *Let G be an undirected graph and $C_g(G)$ be a maximal clique obtained by the greedy heuristic in Figure 21 and $\omega(G)$ denote the cardinality of a maximum clique in G. It is NP-hard to decide if $\omega(G) - |C_g(G)| > 0$ unless $P = NP$.*

Input: An undirected graph G
Output: A maximal clique, C in G

Find the vertex in G with maximum degree, v_{max}

$C = \{v_{max}\}$

Repeat {
  $S = \{w \in V : w \in N(u) \forall u \in C\}$
  for $\forall v \in S$ **do**
    $score(v) = deg_{G[S]}(v)$
  end for
  Let w_{max} be the vertex with maximum score, $C = C \bigcup \{w_{max}\}$
} **until** $S = \emptyset$

Return C

Fig. 21 Greedy heuristic for the maximum clique problem.

Proof. Suppose that we have a polynomial time algorithm $\mathcal{A}_{C_g}(G)$ that, given a graph G , correctly answers the question "Is $\omega(G) - |C_g(G)| > 0$?" with either "yes" or "no". Now let's analyze the case when the output of this algorithm is "no". Let G_i denote the graph obtained from G by adding i vertices that form a clique and are connected to all the vertices in $C_g(G)$. Then we run the following polynomial-time algorithm to compute $\omega(G)$:

0. $G' = G,\ i = 1$;

1. while the answer of $\mathcal{A}_{C_g}(G')$ is "no" repeat

   $G' = G_i$;

   $i = i + 1$.

2. return $i - 1 + C_g(G)$.

Note that each iteration of this algorithm will increase $C_g(G')$ by 1 without changing $\omega(G) = \omega(G')$. The algorithm will terminate when $i-1+C_g(G) = C_g(G') = \omega(G') = \omega_(G)$. By construction of the graph G_i, if v_{max} is a maximum degree vertex in G, it also has the maximum degree in G_i. The same holds for the subsequent construction of the greedy solution. Each time we include a vertex v in the greedy solution whenever it has the largest degree in the induced graph $G[S]$, where S is the set of the common neighbors of the vertices in the current clique. It is easy to see that if v has the largest degree in $G[S]$, then v also has the largest degree in $G_i[S_i]$, where S_i is the set of common neighbors in the current clique in G_i and we have $S_i = S \bigcup G_i \setminus G$. Thus, the existence of such an algorithm would enable us to compute the clique number in polynomial time. □

Proposition 5 implies that one cannot find a polynomial-time approximation algorithm that always gives a solution better than the greedy heuristics outlined in Figure 21 whenever a better solution exists. Since the same greedy heuristic provides a maximal independent set when applied on the complement graph, this result also holds for maximum independent set problem.

VI.4. Minimum Vertex Coloring and Minimum Clique Partitioning

The chromatic number of a graph G is denoted by $\chi(G)$. The well known greedy algorithm for vertex coloring considers the vertices in a specific order $v_1, \ldots, v_n$ and assigns to v_i the smallest available color not used by the neighbors of v_i among $v_1, \ldots, v_{i-1}$, adding a new color if needed. The quality of the resulting coloring depends on the chosen ordering. If the vertices are ordered with respect to their degrees (maximum degree first), the greedy method outputs $\Delta + 1$ colors in the worst case.

Brook's Theorem: $\chi(G) \leq \Delta(G)$ for a connected, simple graph G, unless G is a complete graph or an odd cycle.

Since we can determine whether a graph is a complete graph or an odd cycle in polynomial time, we can easily check if $\chi(G) \leq \Delta(G) + 1$. Thus, the greedy coloring that orders the vertices with respect to degrees cannot be claimed a provably "best" heuristic. Now let us consider ordering with respect to a breadth first search (BFS) tree. We consider connected graphs. It is easy to see that, if the graph is not connected we can run the same procedure for each connected component of the input graph. Suppose G is the input graph. We can assume that G is neither complete nor an odd cycle. Therefore, there exists a vertex v in G with degree less than $\Delta(G)$. We make a breadth first search of G starting at vertex v and label the vertices with respect to the order of their appearance in the search tree. Now that we have an ordering $v_1, \ldots, v_n$, we start coloring the nodes in the backward order. It is easy to see that for any vertex v_i, there exist at most $\Delta(G) - 1$ neighbors already colored. For internal nodes, this is true since they need to have at least one neighbor that has not been colored yet, which is encountered in the search tree before the current one. For the vertex v, this is true since it is not a maximum degree vertex. Thus, we need at most $\Delta(G)$ colors for the whole graph. Now we analyze the theoretical performance of this heuristic based on BFS.

Proposition 6. *Given a connected graph G which is neither complete nor an odd cycle, it is NP-hard to decide whether $\chi(G) < \Delta(G)$.*

Proof. This proposition is easy to prove based on the following results. Maifray and Preissmann [65] proved that the 3-colorability is NP-complete even restricted to the class of triangle-free graphs with maximum degree four. They also state a theorem

derived from the results of Garey and Johnson by using the transformations given in Theorems 4.2 and 4.1 in [39]. This theorem states that the 3-colorability is NP-complete even when restricted to planar graphs with maximum degree four. Although not mentioned explicitly in any of these papers, they directly imply that it is NP-complete to decide whether $\chi(G) < \Delta(G)$ since this is the case even for a planar graph with maximum degree four or a triangle-free graph with maximum degree four. □

Thus, Proposition 6 implies that the greedy coloring based on the order mentioned above, which uses a breadth first search, is a provably "best" heuristic in the sense that it is impossible to have an approximation scheme which always guarantees a better solution whenever it exists, unless $P = NP$.

Since the minimum clique partition problem is equivalent to the minimum vertex coloring in the complement graph, this result also applies to the minimum clique partition problem.

VI.5. The Maximum k-Plex Problem

Given a graph G and an integer $k > 0$, a subset of vertices S is said to be a k-plex if the following condition holds:

$$deg_{G[S]}(v) \geq |S| - k \; \forall v \in S.$$

A k-plex is maximal if it is not strictly contained in any other k-plex. The cardinality of the maximum k-plex is denoted by $\omega_k(G)$. Balasundaram et al. proved the NP-hardness of the maximum k-plex problem for any fixed positive integer k. Similar to the k-club problem, for $l < k$ we have

$$\omega_l(G) \leq \omega_k(G).$$

Proposition 7. *Let positive integers k and l, $l < k$ be given. The problem of checking whether $\omega_l(G) = \omega_k(G)$ is NP-hard..*

Proof. It is easy to prove Proposition 7 by using a similar scheme as in the proof of Proposition 4 for the k-club problem. Assume that there is a polynomial time algorithm $\mathcal{A}_{kl}(G)$ that, given a graph G , correctly answers the question "Is $\omega_l(G) = \omega_k(G)$?" with either "yes" or "no". If the answer is "no", we can compute the value of $\omega_k(G)$ in polynomial time by repeatedly running $\mathcal{A}_{kl}(G)$ until we get a "yes" answer. Similar to the k-club case, at the i^{th} iteration the input graph for $\mathcal{A}_{kl}$ consists of the original graph and a separate clique of size i. Starting from $i = \omega_l(G)$, the $\omega_l(G')$ increases by one at each iteration while $\omega_k(G')$ does not change. Upon termination we have $i = \omega_k(G)$. In case the answer is "yes", we do the following:

0. $G' = G,\ i = 0$;
1. while the answer of $\mathcal{A}_{kl}(G)$ is "yes" repeat

   $G' = G' \cup C_i^{k+2}$;

   $i = i + 1$.
2. return $i + k$.

The graph C_i^{k+2} is constructed as follows. First we form a cycle consisting of $k + 2$ vertices. Clearly this is a k-plex but not an l-plex. Then we add a clique of size i such that every vertex in the clique is connected to all $k + 2$ vertices in the cycle. Thus C_i^{k+2} is a k-plex of size $i + k + 2$ but not an l-plex. Thus, upon termination we observe that $\omega_k(G) = \omega_l(G) = i + k$. The proof is completed by the fact that it is NP-hard to compute $\omega_k(G)$ for any fixed positive integer k.

□

Now we investigate the performance of a greedy heuristic. It is easy to see that any maximal clique can be extended to a maximal k-plex. Now consider the following simple greedy heuristic outlined in Figure 22. We first find a maximal clique by the greedy heuristic described earlier. Next we order the remaining vertices in decreasing order with respect to their degrees. Following this order, we include a vertex in the solution if its inclusion does not violate the k-plex property. Since we search through all the vertices in the graph, the output is a maximal k-plex.

Input: An undirected graph G and an integer $k > 0$
Output: A maximal k-plex, C_k in G

Find a maximal clique C by using the greedy heuristic in Figure 21

$S = V(G) \setminus C$

Order the vertices in S, $v_1, ..., v_{|S|}$, with respect to their degrees in G

$C_k = C$

for $v = v_1 : v_{|S|}$ **do**
    if $C \bigcup \{v\}$ is a k-plex **then**
        $C_k = C_k \bigcup \{v\}$

end for

Return C_k

Fig. 22 Greedy heuristic for the maximum k-plex problem.

Proposition 8. *Let G be an undirected graph and $C_k(G)$ be a maximal k-plex obtained by the greedy heuristic in Figure 22. It is NP-hard to decide if $\omega_k(G) - |C_k(G)| > 0$ unless $P = NP$.*

Proof. Similar to the proof of Proposition 5, suppose we have a polynomial time algorithm $\mathcal{A}(G)$ that correctly answers the question "Is $\omega_k(G) - |C_k(G)| > 0$?" with

either "yes" or "no". If the answer is "no", at the i^{th} iteration we form a clique of size i and connect all the vertices in this clique to all the vertices of $C_k(G)$ and thus obtain the graph G_i. Since the inclusion of these vertices in the graph preserves the degree orders that are used in the greedy heuristics for the maximal clique and maximal $k-$plex, the output of the heuristic at the i^{th} step is $|C_k(G_i)| = |C_k(G)| + i$. Also, observe that $\omega_k(G) = \omega_k(G_i)$. We terminate when the answer of $\mathcal{A}(G)$ is "yes", at which point we are able to identify $\omega_k(G) = \omega_k(G_i) = |C_k(G)| + i$. Thus, by the NP-hardness of the maximum k-plex problem, we conclude that deciding if $\omega_k(G) - |C_k(G)| > 0$ is NP-hard. □

VI.6. Node Deletion Problem

So far we have shown that the greedy heuristics for the maximum clique, maximum independent set, minimum vertex color, minimum clique partition, maximum k-plex and maximum k-club problems are provably "best" in terms of showing the NP-hardness of whether there exists a gap between the heuristic output and the optimal solution. In all these, we used similar arguments except for the minimum vertex coloring, maximum k-club and the minimum clique partition problems. Thus, we expect to have similar results for a broader class of problems that includes all of them except for the vertex color, clique partition and k-club problems.

If π is a graph property, the general node deletion problem can be stated as follows: Find a minimum number of nodes, whose deletion results in a subgraph satisfying property π. We focus on *nontrivial*, *additive* (or *co-additive*), *hereditary* and *interesting* graph properties. A graph property is *nontrivial* if it is true for infinitely many graphs and false for infinitely many graphs. An alternative description for a *nontrivial* property is that it is true for a single node and is not satisfied by all the

graphs in a given input domain. A graph property is *hereditary* on induced subgraphs if for any graph G with property π, all vertex-induced subgraphs of G also satisfy property π. A graph property is *additive*, if it is closed under disjoint unions. It is *co-additive*, if its complement is additive. Some examples of such hereditary properties are planar, outerplanar, bipartite, acyclic, degree constrained, clique (co-additive) and independent set [61]. A property is called *interesting* (in a given input domain) if there are arbitrarily large graphs satisfying π. Observe that neither minimum vertex color nor minimum clique partition problems are related to these graph properties. For the k-club problem, it is easy to see that it is not hereditary on induced subgraphs since deleting a vertex may increase the diameter.

Theorem 18 (Yannakakis [61])**.** *The node-deletion problem for nontrivial, interesting graph properties that are hereditary on induced subgraphs is NP-complete.*

Yannakakis proves the above theorem by a reduction from the vertex cover problem. It is also stated that any ϵ-approximate algorithm for node deletion problems can be used as an ϵ-approximate algorithm for the vertex cover problem. From now on we focus on the complement of the node deletion problem. The *maximum subgraph with property π problem* is to find a maximum cardinality subgraph S in G such that S satisfies property π. Let S_o be the optimal solution for this problem and D_o be the optimal solution for the node deletion version for the same property π. Then, we have: $S_o \bigcup D_o = V(G)$. Thus the *maximum subgraph with property π problem* is also NP-hard. Observe that with the additional restriction of being "additive" or "co-additive", this problem is still NP-hard.

Conjecture 1. Let $\mu_\pi(G)$ denote the size of the largest subset of vertices in G satisfying the property π and let $\mu_\pi^g(G)$ denote the size of a maximal by inclusion subset of vertices satisfying the property π that is computed using a simple greedy

heuristic which considers a certain ordering of vertices. Then recognizing whether $\mu_\pi(G) > \mu_\pi^g(G)$ is NP-hard.

Suppose that we have a maximal by inclusion greedy which considers the vertices in a certain order, i.e with respect to their degrees. Similar to the proofs of maximum clique and maximum k-plex problems, we would like to construct a graph G_i with some additional vertices such that $\mu_\pi^g(G_i) = \mu_\pi^g(G) + i$ and $\mu_\pi(G) = \mu_\pi(G_i)$ and also G_i does not alter the order of vertices for the greedy in G. The order is important to guarantee that the heuristic solution for G_i is the union of the heuristic solution for G and the set $G_i \setminus G$. The existence of such a graph G_i guarantees that if we had an algorithm $\mathcal{A}(G)$ that answers if $\mu_\pi(G) > \mu_\pi^g(G)$ in polynomial time, we could run that algorithm repeatedly till we obtain $\mu_\pi(G) = \mu_\pi(G_i) = \mu_\pi^g(G_i)$. Thus, we would be able to compute $\mu_\pi(G)$ in polynomial time which contradicts with the fact that it is NP-hard. Therefore, we need to prove that we can always construct such a graph G_i for this class of problems.

The hereditary property is the key in being able to design a maximal by inclusion greedy. For a given nontrivial, hereditary and interesting property π, suppose that the optimal solution for the *maximum subgraph with property π problem* is $\pi(G)$ and the greedy maximal by inclusion solution is $\pi_g(G)$. Without loss of generality, we can assume that $\pi(G) \neq V(G)$ since in this case we can easily verify optimality because we would have $\pi_g(G) = V(G) = \pi(G)$. Thus, if the answer of $\mathcal{A}(G)$ is "no", there exists at least one vertex v in G such that the set $\pi_(G) \bigcup \{v\}$ does not have the property π. Therefore, it is possible to create some graph G_i such that the additional i vertices do not alter the optimal solution but we have to make sure that it also satisfies the condition on the heuristic solution.

Lemma 16 (Yannnakakis [61])**.** *Let π be any graph property that is hereditary, non-*

trivial and interesting. Then either all cliques or all independent sets of nodes satisfy π.

Proof. This proof is by Yannakakis [61]. He uses Lemma 16 to prove the NP-hardness of node deletion problems in [61]. For all m, n there is a number $r(m, n)$ (Ramsey number), such that every graph with no fewer than $r(m, n)$ nodes contains either a clique of m nodes, K_m, or an independent set of n nodes, $\overline{K}_n$. Suppose that the statement in the lemma is not true. Thus there are m, n such that K_m and $\overline{K}_n$ does not satisfy π. Since π is a nontrivial property there is a graph satisfying π, with more than $r(m, n)$ nodes. Since π is hereditary on induced subgraphs, either K_m or $\overline{K}_n$ has to satisfy π. □

As pointed out by Yannakakis, we can always define a complementary property $\overline{\pi}$ as follows: A graph G satisfies $\overline{\pi}$ if and only if its complement $\overline{G}$ satisfies π. It is easy to see that $\overline{\pi}$ is also hereditary, nontrivial and interesting whenever π is. Now we can assume that all independent sets of nodes satisfy π and π is additive; otherwise we consider the equivalent problem for $\overline{\pi}$. Next we consider the following graph $G_{i,w}$:

$$\begin{aligned} V(G_{i,w}) &= V(G) \bigcup V(I_i) \\ E(G_{i,w}) &= E(G) \bigcup \{(uv) : u \in I_i, v \in N_{[V(G)]}(w) \setminus \pi_g(G), \forall u, \forall v\} \end{aligned}$$

where I_i denotes an independent set of size i and w is a vertex in $\pi_g(G)$. We connect each vertex of I_i to each vertex of $N_{[V(G)]}(w) \setminus \pi_g(G)$.

Lemma 17. *Let G be a graph such that it satisfies a nontrivial, hereditary, additive and interesting property π. If all independent sets of nodes satisfy π, then $G \bigcup I$ also satisfy π where I is an independent set.*

Proof. Since all independent sets satisfy property π, by the *additive* rule $G \bigcup I$ also satisfies property π since $G \bigcup I$ is a disjoint union. □

Now consider the following algorithm displayed in Figure 23: If the answer of $\mathcal{A}(G)$ is "no", we create graphs $G' = G_{i,w}$ for each $w \in \pi_g(G)$. Before incrementing i, we run the algorithm $\mathcal{A}(G')$ for each $w \in \pi_g(G)$. If none of the answers is "yes", we increment and repeat the same procedure.

Input: An undirected graph G and an interesting, nontrivial, additive (or co-additive) and hereditary property π
Output: Maximum subgraph with property π

Compute $\pi_g(G)$

$G' = G$, $i = 0$

while the answer of $\mathcal{A}(G')$ is "no" **repeat**
  $i = i + 1$
  for each $w \in \pi_g(G)$ **do**
    $G' = G_{i,w}$
    if $\mathcal{A}(G')$ is "yes"
    Return $\pi_g(G) + i$
  end for
end while

Fig. 23 General solution scheme for node deletion problems based on heuristics.

Conjecture 2. For the algorithm displayed in Figure 23, there exists a vertex $w \in \pi_g(G)$ such that $\mu^g_\pi(G_{i,w}) = \mu^g_\pi(G) + i$, $\mu_\pi(G) = \mu_\pi(G_{i,w})$ and the heuristic order of the vertices in $\pi_g(G)$ is preserved.

Based on Lemma 17, the property $\mu^g_\pi(G_{i,w}) = \mu^g_\pi(G) + i$ is satisfied by all the graphs $G_{i,w}$. Furthermore, by construction, the degrees of the vertices in $\pi_g(G)$ do not change while the degrees of the vertices in $G \setminus \pi_g(G)$ increase by one at each iteration. Thus, if we order the vertices starting with minimum degree first, the order of the original vertices does not change. It is easy to see that the order of the added vertices

does not alter the greedy solution. If they are in the beginning, they can be only extended to $\pi_g(G) + I_i$. If they come afterwards, the output will be still $\pi_g(G) + I_i$. Thus, we need to prove that $\mu_\pi(G) = \mu_\pi(G_{i,w})$ for some vertex $w \in \pi_g(G)$. Observe that if there exists a vertex $w \in \pi_g(G)$ such that it is not adjacent to any vertices in $\pi_g(G) \bigcap \pi(G)$, then we have the desired property.

Conjecture 3. There exists a vertex $w \in \pi_g(G)$ such that $N(w) \bigcap \pi_g(G) \bigcap \pi(G) = \emptyset$.

The proof of Conjecture 3, thus Conjecture 2, will imply the proof of Conjecture 1 in the special case where π or $\overline{\pi}$ is additive and satisfied by all independent sets.

When proven, Conjecture 1 can be used as a theoretical justification for the choice of greedy heuristics for several optimization problems such as maximum planar induced subgraph, maximum outerplanar induced subgraph, maximum bipartite induced subgraph, maximum clique, maximum k-plex and maximum independent set problems.

VI.7. Conclusion

In this chapter, we focused on the theoretical performance of greedy heuristics for several problems. Simple greedy heuristics are often used to solve large-scale instances of NP-hard problems in practice. They are mostly easy to understand and implement. However, there is a fair amount of skepticism towards such approaches due to a lack of theoretical foundations behind them. Many greedy heuristics even do not have a provable approximation ratio. Thus, a reasonable question is "why do we choose a certain greedy heuristic for our solution procedure". Inapproximability results for several optimization problems are well-known. When an optimization problem is said to be not approximable within a factor of some constant $c - \epsilon$, it is easy to claim

that if an approximation algorithm has a performance guarantee of c, it is the "best" possible. However, for many problems such as maximum clique, the inapproximability result is in terms of the problem size. Thus, we can not claim a heuristic to be the "best" with respect to its approximation ratio. Motivated by this problem, we propose a method for the theoretical justification for the choice of heuristics.

We considered several optimization problems such as maximum k-club, maximum k-clique, maximum clique, maximum independent set, maximum k-plex, minimum vertex coloring and minimum clique partitioning. For each one of these problems, we showed that a simple greedy heuristic is proven to be the "best" in the sense that it is NP-hard to decide whether there is a gap in between the heuristic solution and the optimal solution. The existence of a polynomial time algorithm that always guarantees a solution better than the simple heuristic would mean that we can check the greedy heuristic-optimal gap in polynomial time.

We also worked on extensions of our results for a more general class of graph problems. The *maximum subgraph with property π problem* is finding a subgraph of maximum cardinality that satisfies some given property π. We focused on properties that are hereditary on induced subgraphs, nontrivial and interesting. For such properties, the *maximum subgraph with property π problem* is NP-hard. We conjectured that any maximal by inclusion greedy heuristic based on some order of vertices is "best" in the sense that it is not possible to have a polynomial time approximation algorithm which always gives a better solution. This class includes several optimization problems some of which are maximum planar induced subgraph, maximum bipartite induced subgraph, maximum clique, maximum outerplanar induced subgraph and maximum degree-constrained subgraph problems. We presented some arguments that can be used in proving this conjecture.

Our results may also be viewed as an additional evidence of these problems'

practical intractability. On the other hand, this should not prevent the practitioners from designing more sophisticated approaches for these problems, since the above result describes just the worst-case behavior of the heuristics.

CHAPTER VII

GREEDY CONSTRUCTION HEURISTICS FOR THE MAX-CUT PROBLEM*

VII.1. Introduction

Given an undirected graph $G = (V, E)$ with the set of vertices $V = \{1, 2, 3, \ldots, n\}$ and the set of edges E with weights $w_{ij} \geq 0$ for each $(i, j) \in E$, the max-cut problem is to find a partition of vertices into two disjoint subsets S_1 and S_2 such that the sum of weights of the edges with endpoints in different subsets is maximized. Each partition of vertices into two subsets S_1 and S_2 with $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = V$ is called a cut, and the total weight of edges with endpoints in different subsets is called the weight of the cut or the cut value and is denoted by $cut(S_1, S_2)$:

$$cut(S_1, S_2) = \sum_{i \in S_1, j \in S_2} w_{ij}.$$

The max-cut problem finds applications in statistical physics and circuit layout design [16]. Other applications include social networks, where the max-cut value is generally a measure of robustness of the network [17, 4], and classification [25].

Like many other graph theory problems, max-cut is very easy to state but hard to solve. It is a well known NP-hard problem [57]. The max-cut problem can be

*Reprinted with permission from "On greedy construction heuristics for the MAX-CUT problem" by S. Kahruman, E. Kolotoglu, S. Butenko and I. V. Hicks, 2007. International Journal of Computational Science and Engineering, Volume Number 3, 211-218, Copyright © 2007 Inderscience Enterprises Ltd.

formulated as the following mixed integer linear program:

$$\max \sum_{\substack{i,j=1 \\ i<j}}^{n} w_{ij} y_{ij}, \tag{7.1}$$

subject to:

$$y_{ij} - x_i - x_j \leq 0,\ i, j = 1, 2, \ldots, n,\ i < j; \tag{7.1a}$$

$$y_{ij} + x_i + x_j \leq 2,\ i, j = 1, 2, \ldots, n,\ i < j; \tag{7.1b}$$

$$x_i \in \{0, 1\},\ i = 1, 2, \ldots, n. \tag{7.1c}$$

In the above, an optimal solution (x^*, y^*) corresponds to an optimal max-cut partition of V into two subsets $S_1 = \{i : x_i^* = 0\}$ and $S_2 = \{i : x_i^* = 1\}$. A detailed polyhedral study of the problem is given in [31]. Note that the mixed integer formulation that we propose above has the same number of integer variables as the number of vertices in the graph, while the known integer programming formulations, including that proposed in [31], have a quadratic number of integer variables with respect to the number of vertices. However, the above formulation also has a quadratic number of non-integer variables in addition to the integer variables. We will also mention the following nonconvex quadratic formulation [11]. The optimal objective function value of max-cut problem is given by

$$\max_{x \in [0,1]^n} x^T W (e - x), \tag{7.2}$$

where $W = [w_{ij}]_{i,j=1}^n$ is the matrix of edge weights (with zero diagonal), and $e = [1, 1, \ldots, 1]^T$ is the unit vector of length n. Similar quadratic formulations with binary variables are typically used to obtain semidefinite programming relaxations for the max-cut problem and derive approximation algorithms and heuristics based on such relaxations [20, 19, 41, 64].

The known polynomially solvable cases include planar graphs [44], graphs without K_5 minors [15], and weakly bipartite graphs with nonnegative weights [43]. max-cut on dense graphs [78] and metric max-cut have polynomial time randomized approximation schemes [79]. However, metric max-cut is not known to be NP-hard. The general version of max-cut problem is also known to be APX-complete [68], meaning that unless P=NP, it does not allow a polynomial time approximation scheme (PTAS) [77]. Thus, approximation algorithms or heuristics are used for finding acceptable solutions in polynomial time.

In this paper, we compare the performance of several greedy construction heuristics for the max-cut problem. In particular, we present and study a new "worst-out" construction approach for the max-cut problem, the *edge contraction heuristic*. We show that the proposed algorithm has the approximation ratio of at least 1/3. We also present an experimental comparison of solutions obtained using the edge contraction heuristic, the classical "best-in" $\frac{1}{2}$-approximation algorithm of Sahni and Gonzales [70], and modifications for both.

The remainder of this paper is organized as follows. Section VII.2 briefly surveys the known construction algorithms for max-cut problem. The edge contraction heuristic is proposed and analyzed in Section VII.3. Some modified versions of the Sahni-Gonzalez algorithm are presented in Section VII.4. Section VII.5 presents the results of experimental comparison of several greedy construction heuristics for the problem of interest in terms of the solution quality. Finally, some concluding remarks are given in Section VII.6.

VII.2. Construction Algorithms and Approximation Ratios

Most heuristic approaches to the max-cut problem consist of a construction algorithm, which builds a feasible solution from scratch, and a local search procedure that attempts to iteratively improve the current solution until a local maximum with respect to a given neighborhood is reached. In this paper we deal only with construction algorithms. More specifically, we are interested in simple greedy construction approaches and their variations. There are two major types of greedy construction algorithms for discrete optimization problems on graphs: "best in" and "worst out". A best-in algorithm typically starts with an empty graph (or a very small subgraph of the input graph), while various subgraphs of the original graph (which may be a vertex or an edge, depending on the problem) are considered to be candidates for inclusion in the constructed feasible solution. Then the algorithm successively adds a candidate, which provides the "best" contribution to the objective function value, and removes the candidates that become ineligible for inclusion from the list of candidates. The procedure is repeated until a feasible solution is constructed (for minimization problem) or the candidate list is empty. Alternatively, a worst-out algorithm usually starts with the input graph and on each step removes the part of the graph (such as a vertex or an edge), which, if included in the solution, would provide a "worst" contribution to the objective function value compared to all other candidates for removal. The algorithm stops when the remaining graph constitutes a feasible solution (for maximization problem) or any additional step would make otherwise feasible graph infeasible. Note that this description of best-in and worst-out algorithms aims to provide a general idea behind such algorithms and does not intend to be restrictive, as numerous variations of the outlined greedy approaches can still be called best-in or worst-out algorithms. This paper was partially motivated by observation that, while

there are several well-known best-in algorithms for max-cut problem, no results on worst-out approaches have been published to the best of our knowledge. In particular, we are interested in comparing the the proposed worst-out and the known best-in algorithms in terms of their approximation ratios and quality of the solutions obtained in numerical experiments. Next we define the concept of approximation ratio for a max-cut algorithm and mention some of the known algorithms for max-cut and their approximation ratios.

Let $W_{\mathcal{A}}(G)$ be the cut size generated by an approximation algorithm $\mathcal{A}$ for max-cut problem on a graph G. The approximation ratio of the algorithm $\mathcal{A}$ is defined as the largest $R_{\mathcal{A}}$ for which

$$W_{\mathcal{A}}(G)/W^*(G) \geq R_{\mathcal{A}} \text{ for any graph } G,$$

where $W^*(G)$ is the optimal cut value of G. Note that $W^*(G) \leq W(G)$, where $W(G) = \sum_{i<j} w_{ij}$ is the sum of all edge weights of the graph. Thus, any R such that $W_{\mathcal{A}}(G)/W(G) \geq R$ for any G provides a lower bound on $R_{\mathcal{A}}$. Since finding a better than $W(G)$ upper bound on the size of maximum cut may be nontrivial, this bound is frequently used to estimate an algorithm's approximation ratio.

In 1976, Sahni and Gonzalez [70] presented an algorithm that constructs an approximate solution to max-cut with the approximation ratio of 1/2. The time complexity of this algorithm is $O(|V| + |E|)$. Their algorithm starts by placing one vertex to each partition, and the remaining $|V| - 2$ vertices are examined one by one. A vertex j is assigned to a partition if the total weight of the edges in between vertex j and the vertices in that partition is minimal. This algorithm, which is perhaps the first known approximation algorithm for max-cut, is still quite popular due to its simplicity and reasonably good quality of solution it guarantees. Recently, Festa et al. [37] implemented and tested a greedy randomized adaptive search procedure, a

variable neighborhood search and a path-relinking intensification heuristic for max-cut. In the construction phase, at each iteration, an element is randomly selected from a restricted candidate list, whose elements are ranked according to the main idea that Sahni and Gonzalez used.

Haglin and Venkatesan [45] proposed an algorithm that guarantees an approximation ratio of at least of $1/2 + \frac{1}{2|V|}$ starting with a matching of size $|E|/|V|$. Their algorithm runs in $O((|E| \log |E| + |V| \log |V|)/p + \log p \log |V|)$ parallel time using $1 <= p <= |E| + |V|$ processors. They also showed that it is NP-Complete to decide if a given graph has a maximum cut with at least a fraction $1/2 + \varepsilon$ of the sum of weights of its edges, where ε is a positive constant. Cho et al. [27] proposed an improved approximation algorithm running in $O(|E| + |V|)$ sequential time yielding a node-balanced maximum cut with size at least $W(G)(1/2 + 1/2|V|)$. Although the approximation ratio is the same as Haglin and Venkatesan's, their algorithms is better in terms of time complexity. They initialize the partitions to be empty and find a matching M of size $|E|/|V|$ to be included in the final cut. Then, they assign the vertices to partitions considering a vertex pair at a time such that the cut value in between the partitions is maximized. Kajitani et al. [56] modified the Haglin and Venkatesan's approach by using a matching with $|E|/(|V|-1)$ edges in G, which they computed in $O(|E| + |V|)$ time. This allowed them to obtain an approximation ratio of $1/2 + \frac{1}{2(|V|-1)}$ which is a slight (but not asymptotic) improvement.

The most remarkable approximation results for max-cut problem are associated with using semidefinite programming relaxations of max-cut formulations. In their breakthrough paper [41], Goemans and Williamson used semidefinite programming to develop an algorithm for max-cut that always delivers solutions of expected value at least 0.87856 times the optimal value. Feige et al. [58] improved the last step of the Goemans-Williamson algorithm to obtain an approximation ratio of at least 0.921

for graphs of maximum degree three. Liu et al. [64] proposed a tighter semidefinite relaxation of max-cut. For cubic graphs, *i.e.*, graphs in which the degree of all vertices is three, Halperin et al. [48] presented an improved semidefinite programming based approximation algorithm, which has an approximation ratio of 0.9326. Semidefinite programming approaches yield algorithms with the best known approximation ratios, however, the time and space requirements limit their applicability in practice.

In 2002, Alperin and Nowak presented a smoothing heuristic based on Lagrangian relaxation [7]. The heuristic is based on a parametric optimization problem defined as a convex combination between a Lagrangian relaxation and the original problem. Starting from the Lagrangian relaxation, a path following method is applied to obtain good solutions while gradually transforming the relaxed problem into the original problem formulated with an exact penalty function.

Although researchers found improvements, especially by making use of semidefinite programming, since the publication of the very basic 0.5-approximation algorithm of Sahni and Gonzalez, there has not been much progress in developing algorithms with a constant approximation ratio that would be fast, simple and effective in practice. This is especially important for the cases where one needs to solve max-cut as a subroutine many times. It should also be noted that as long as total weight $W(G)$ of all edges in the graph is used instead of the optimal cut value in derivation of the approximation ratio result, we cannot prove that the approximation ratio is better than $1/2$ for any algorithm. This is because there exist graphs on which the max-cut value is very close to $W(G)/2$. At the same time, finding a tighter upper bound for the max-cut value is not trivial. In fact, this bound is sharp in some sense, since a bipartite graph has a max-cut value which is exactly $W(G)$. The need for improved simple algorithms which would outperform the algorithm proposed by Sahni and Gonzalez in terms of solution quality in practical applications is another motivation to consider

several alternative greedy approaches which we present and analyze in the following two sections.

VII.3. The Edge Contraction Heuristic

Let $e = (x, y)$ be an edge of a graph $G = (V, E)$. Contracting an edge e means forming a new vertex $v = v_{xy}$ out of e, which becomes adjacent to all the former neighbors of x and y. The edge contraction heuristic takes the graph $G = (V, E)$ as an input. If the graph is not complete, add all missing edges with weight zero to the original graph. The minimum weighted edge of G is contracted and the graph is updated. Each time an edge e is contracted, the number of the vertices in the graph decreases by one. This procedure is done repeatedly until the number of vertices remaining in the graph becomes two. This heuristic is a worst-out greedy method. The motivation for this method comes from the fact that at each iteration, contracting the minimum weighted edge corresponds to removing this edge from the final solution by assigning the adjacent vertices to the same partition.

In the algorithm, whenever an edge e, whose endpoints are the vertices x and y, is contracted, we form the edges adjacent to the new vertex $v = v_{xy}$. Let i be a vertex distinct from x and y. Then the weight of the new edge between the vertices v_{xy} and i is obtained by adding the weights of the edges (x, i) and (y, i): $w_{vi} = w_{xi} + w_{yi}$. At each step, each vertex v has a *contraction list* which contains all the vertices adjacent to edges that were contracted in previous steps of the algorithm to form vertex v. When the algorithm stops, we have two vertices whose *contraction lists* give us the output cut partition. The steps of the algorithm are summarized in Figure 24. It is easy to see that the time complexity of this algorithm is $O(|V|^3)$.

Note that if we use $|V| - k$ steps instead of $|V| - 2$ steps in the main **for**-loop of

Input: A complete graph $G(V, E)$ with edge weights $w_{ij}, \forall i, j \in V,\ i \neq j$
Output: A cut $S_1, S_2 : S_1 \cup S_2 = V, S_1 \cap S_2 = \emptyset$ and the cut value $cut(S_1, S_2)$

for $j = 1 : |V|$ **do**
    $ContractionList(j) = \{j\}$
end

for $j = 1 : |V| - 2$ **do**
    Find a minimum weight edge (x, y) in G
    $v = contract(x, y)$
    $V = V \cup \{v\} \setminus \{x, y\}$
    for $i \in V \setminus \{v\}$ **do**
        $w_{vi} = w_{xi} + w_{yi}$
    end
    $ContractionList(v) = ContractionList(x) \cup ContractionList(y)$
end

Denote by x and y the only 2 vertices in V
    $S_1 = ContractionList(x)$
    $S_2 = ContractionList(y)$
    $cut(S_1, S_2) = w_{xy}$

Return $cut(S_1, S_2)$

Fig. 24 The Edge Contraction Heuristic.

this algorithm, then we obtain a heuristic for the maximum k-cut problem, which is to partition all vertices into k disjoint sets so that the some of the weights of all edges with endpoints in different parts is maximized. The $ContractionList(i)$ set of each remaining vertex i would correspond to the partitions, while the objective function value would be the sum of weights of all remaining edges.

The following lemma will be used to prove an approximation ratio result for the contraction algorithm for the max-cut problem.

Lemma 18. *Let W_k denote the total weight of the first k edges contracted by the edge contraction heuristic and let W be the total weight of all edges in the graph. Then for any $1 \leq k \leq n-2$*

$$W_k \leq \frac{2kW}{(n-1)(n-k+1)}$$

where n is the number of vertices in the input graph.

Proof. The proof is based on the fact that the weight of a contracted edge is no greater than the average edge weight in the current graph at each iteration. This is true since the procedure always contracts an edge of the smallest weight. So, the weight of the first contracted edge satisfies

$$W_1 \leq \frac{W}{\binom{n}{2}} = \frac{2W}{n(n-1)}.$$

We will use induction on k. We have already shown that the lemma is valid for $k = 1$. Assume it is correct for all integer $k \leq \kappa$. We need to derive the inequality in lemma for $k = \kappa + 1$. Expressing the upper bound on $W_{\kappa+1}$ through W_κ and W, and using the induction assumption for W_κ, we obtain:

$$\begin{aligned}
W_{\kappa+1} &\leq W_\kappa + \frac{W - W_\kappa}{\binom{n-\kappa}{2}} = \frac{(n-\kappa)(n-\kappa-1)-2}{(n-\kappa)(n-\kappa-1)} W_\kappa + \frac{2W}{(n-\kappa)(n-\kappa-1)} \\
&\leq \frac{2W}{(n-\kappa)(n-\kappa-1)} \left(\frac{((n-\kappa)(n-\kappa-1)-2)\kappa}{(n-1)(n-\kappa+1)} + 1 \right) \\
&= \frac{2(\kappa+1)W(n-\kappa+1)(n-\kappa-1)}{(n-\kappa)(n-\kappa-1)(n-1)(n-\kappa+1)} \\
&= \frac{2(\kappa+1)W}{(n-1)(n-\kappa).}
\end{aligned}$$

Thus,

$$W_{\kappa+1} \leq \frac{2(\kappa+1)W}{(n-1)(n-\kappa)}$$

and by induction the lemma is correct. □

Theorem 19. *Denote by W_c the value of the cut obtained using the edge contraction heuristic and by W^* the weight of an optimal cut. Then*

$$W_c \geq \frac{1}{3}\left(1 + \frac{2}{n-1}\right) W$$

and, in particular,

$$W_c > \frac{1}{3} W^*.$$

Here, as before, W denotes the total weight of all edges in the graph.

Proof. The proof follows from Lemma 18. Indeed, $W_c = W - W_{n-2}$, thus from the lemma

$$W_c \geq W - \frac{2(n-2)W}{(n-1)(n-(n-2)+1)} = \frac{1}{3}\left(1 + \frac{2}{n-1}\right) W,$$

and since $W \geq W^*$, we obtain $W_c > \frac{1}{3} W^*$. □

VII.4. Modifications to the Edge Contraction Heuristic and Sahni-Gonzalez Algorithm

In this section, we discuss four algorithms which are obtained by modifying the edge contraction heuristic and the Sahni-Gonzalez algorithm. The algorithms SG1, SG2 and SG3 are variations of Sahni-Gonzalez algorithm where the order to consider the vertices depends on a score function.

The compromise heuristic: This heuristic is a combination of the edge contraction heuristic and the Sahni-Gonzalez algorithm. We first apply the edge contraction heuristic until the weight of the minimum weighted edge on updated graph becomes greater than or equal to the average edge weight of the input graph. The intuition behind this idea is that if the edge weights in a graph are much

smaller than the average edge weight, then it is more likely that the endpoints of these type of edges will be in the same partition of the max-cut. We will call them "light edges" and the edges that are not light will be called "heavy". After all light edges are contracted, we apply the Sahni-Gonzalez algorithm on the updated graph.

SG1: This is a best-in algorithm, which is a modification of the Sahni-Gonzalez approach. Its steps are summarized in Figure 25. Here $w(i, S_j)$ denotes the total weight of the edges in between vertex i and the vertices in the partition $V_j,\ j = 1, 2$. At each iteration, the SG1 algorithm considers all the remaining vertices and picks the one which will contribute the most to the current cut value at that iteration.

SG2: The SG2 algorithm is very similar to SG1. The differences are in the definition of the *score* function and in the choice of the next vertex to be included in one of the partitions:

- $score(i) = \min\{w(i, S_1), w(i, S_2)\}$
- Choose the vertex i^* with the minimum score.

This algorithm can be thought of as a best-in algorithm for the minimum 2 set partitioning problem, which is to partition all vertices into two disjoint subsets so that the sum of weights of edges with both endpoints in the same partition is minimized. Obviously, this problem is equivalent to the max-cut problem. Indeed, at each step a vertex that is the best in terms of contributing to the goal of minimizing the current partitions weight is chosen.

SG3: This algorithm is also very similar to SG1, the only difference being the score

Input: A complete graph $G(V, E)$ with edge weights $w_{ij}, \forall i, j \in V,\ i \neq j$
Output: A cut $S_1, S_2 : S_1 \cup S_2 = V, S_1 \cap S_2 = \emptyset$ and the cut value $cut(S_1, S_2)$

0. $V' = V$
   Pick the maximum weighted edge (x, y)
   $cut(S_1, S_2) = w_{xy}$
   $V' = V' \setminus \{x, y\}$
   $S_1 = \{x\}$, $S_2 = \{y\}$
1. **for** $j = 1 : n - 2$ **do**
   for $i \in V'$ **do**
   $score(i) = \max\{w(i, S_1), w(i, S_2)\}$
   end
   Choose the vertex i^* with the maximum score
   If $w(i^*, S_1) > w(i^*, S_2)$ **then** $S_2 \leftarrow S_2 \bigcup \{i^*\}$
   else $S_1 \leftarrow S_1 \bigcup \{i^*\}$
   $cut(S_1, S_2) = cut(S_1, S_2) + score(i^*)$
   end
   Return $cut(S_1, S_2)$

Fig. 25 The SG1 Algorithm.

function. In this case the score of each remaining vertex is calculated as follows:

$$score(i) = |w(i, S_1) - w(i, S_2)|$$

In fact, SG3 can be viewed as a clever combination of SG1 and SG2. For all the vertices, it takes into account the contribution to the minimization of the current partition weight and at the same time the contribution to the current cut value in that iteration by simply looking at the absolute difference.

In the next section, we present the results of the numerical experiments.

VII.5. Numerical Results

This section presents the results of the numerical experiments to compare the performances of the algorithms introduced in Sections 3 and 4, and also the basic Sahni-Gonzalez [70] algorithm. We tested these algorithms on several randomly generated graphs and some TSP instances from TSPLIB. These graphs are all weighted and complete. In addition, some instances were taken from Resende et al. [37] (G1, G2, G15, G17 and G53 are all 0, 1 weighted). The size of the TSP instances are already specified in their names and for an instance named rxx, yy, xx denotes the number of the vertices and yy denotes the maximum weight of edges. G1, G2, G15 and G17 are of size 800 while G53 is of size 1000. We first compare the results for Sahni-Gonzalez algorithm and the edge contraction heuristic in Table 5. In this table, W_{SG} and W_C represent the value of the cut obtained using the Sahni-Gonzalez and the contraction algorithm, respectively, while W stands for the sum of weights of all edges in the graph.

From Table 5, the Sahni-Gonzalez algorithm outperforms the edge contraction heuristic by giving better approximation in general. In particular, on three instances ($r25, 30$; $r45, 50$; and $r150, 3$) the ratio W_C/W is less than $1/2$, which is the approximation ratio of the Sahni-Gonzalez algorithm. Thus, the approximation ratio of the edge contraction heuristic is less than that of the Sahni-Gonzalez algorithm. The $1/3$ bound derived in Theorem 19 may be tight, but has not been proven so. However, on 5 out of 9 TSP instances, regarded as "dense" instances, the edge contraction heuristic does better than the Sahni-Gonzalez algorithm. This observation enhances our motivation for the compromise heuristic. The Sahni-Gonzalez algorithm is also better than the edge contraction heuristic in terms of running time. The time complexity of the edge contraction heuristic is $O(|V|^3)$. Since we are considering the complete

Table 5 Comparative results of the algorithm of Sahni and Gonzalez [70] and the edge contraction heuristic.

Instance		Sahni-Gonzalez		Edge Contraction	
Name	W	W_{SG}	W_{SG}/W	W_C	W_C/W
$Burma14$	355	193	0.543662	254	0.715493
$gr17$	37346	23354	0.625341	24986	0.669041
$bayg29$	66313	36939	0.55704	35058	0.528675
$bays29$	370530	214339	0.578466	226102	0.610212
$dantzig42$	59574	30508	0.512103	36023	0.604677
$att48$	3.74E+06	2.49E+06	0.665833	2.25E+06	0.602836
$hk48$	1.15E+06	712355	0.617408	732706	0.635046
$berlin52$	762783	453174	0.594106	445739	0.584359
$brazil58$	3.52E+06	1.92E+06	0.543782	1.83E+06	0.519003
$r15,30$	2679	1504	0.561404	1504	0.561404
$r20,30$	8723	4675	0.535939	4522	0.5184
$r25,30$	6115	3384	0.553393	3038	0.496811
$r25,40$	12161	6603	0.542965	6186	0.508675
$r30,40$	15202	8153	0.536311	8073	0.531049
$r35,40$	18926	10293	0.543855	10122	0.53482
$r45,50$	44705	23708	0.530321	21840	0.488536
$r55,40$	29487	16306	0.552989	15827	0.536745
$r55,50$	36889	20375	0.552333	20041	0.543278
$r63,75$	72801	39604	0.544004	39163	0.537946
$r75,130$	177855	96491	0.542526	92445	0.519777
$r80,10$	15737	8492	0.53962	7873	0.50028595
$r82,130$	214256	115466	0.538916	108347	0.505689
$r82,20$	33461	17973	0.537133	16868	0.504109
$r100,50$	122178	65834	0.538837	61851	0.506237
$r150,3$	16863	9041	0.536144	8426	0.499674
$r150,4$	22064	11785	0.534128	11066	0.501541
$r250,100$	1.57E+06	813292	0.519272	796739	0.508703
$r500,101$	6.35E+06	3.27E+06	0.514598	3.23E+06	0.508212
$G1$	19176	10949	0.570974	9794	0.510743
$G2$	19176	11050	0.576241	9955	0.519139
$G15$	4661	2865	0.614675	2645	0.567475
$G17$	4667	2883	0.617742	2656	0.569102
$G53$	5914	3642	0.615827	3352	0.566791

Table 6 Comparative results of the ratios of the cut value to the graph's total edge weight achieved by SG, C, CSG, $SG1$, $SG2$ and $SG3$.

Name	SG	C	CSG	$SG1$	$SG2$	$SG3$
$burma14$	0.543662	0.715493	0.695775	0.721127	0.715493	0.721127
$gr17$	0.625341	0.669041	0.555669	0.669041	0.669041	0.669041
$bayg29$	0.55704	0.528675	0.539306	0.559242	0.569632	0.564173
$bays29$	0.578466	0.610212	0.57330	0.642666	0.617070	0.642666
$dantzig42$	0.512103	0.604677	0.509659	0.677024	0.677024	0.677024
$att48$	0.665833	0.602836	0.672023	0.674483	0.674661	0.674661
$hk48$	0.617408	0.635046	0.614240	0.668855	0.668855	0.668855
$berlin52$	0.594106	0.584359	0.572952	0.616434	0.614527	0.617117
$brazil58$	0.543782	0.519003	0.543635	0.546723	0.556287	0.563592
$r15_{,3}0$	0.561404	0.561404	0.567376	0.565136	0.578201	0.583053
$r20, 30$	0.535939	0.518400	0.529634	0.544079	0.548321	0.550384
$r25, 30$	0.553393	0.496811	0.575470	0.573508	0.569583	0.573181
$r25, 40$	0.542965	0.508675	0.532522	0.555464	0.553326	0.554889
$r30, 40$	0.536311	0.531049	0.543810	0.552296	0.548349	0.552033
$r35, 40$	0.543855	0.534820	0.528796	0.551358	0.550988	0.557117
$r45, 50$	0.530321	0.488536	0.523722	0.542624	0.541461	0.541640
$r55, 40$	0.552989	0.536745	0.541832	0.556856	0.548377	0.563401
$r55, 50$	0.552333	0.543278	0.539863	0.558107	0.548619	0.559869
$r63, 75$	0.544004	0.537946	0.538770	0.552932	0.548797	0.555308
$r75, 130$	0.542526	0.519777	0.539872	0.548160	0.541705	0.555149
$r80, 10$	0.539620	0.500286	0.534981	0.549215	0.543750	0.554299
$r82, 130$	0.538916	0.505689	0.529096	0.539850	0.543285	0.549833
$r82, 20$	0.537133	0.504109	0.533815	0.542004	0.545710	0.548549
$r100, 50$	0.538837	0.506237	0.526781	0.539606	0.539639	0.542299
$r150, 3$	0.536144	0.499674	0.530688	0.538813	0.541481	0.544565
$r150, 4$	0.534128	0.501541	0.522117	0.538751	0.537255	0.540790
$G1$	0.570974	0.510743	0.531237	0.584220	0.580622	0.591834
$G2$	0.576241	0.519139	0.522685	0.586045	0.578796	0.594180
$G15$	0.614675	0.567475	0.541729	0.628835	0.597726	0.642137
$G17$	0.617742	0.569102	0.537819	0.628455	0.599529	0.638097
$G53$	0.615827	0.566791	0.542273	0.629523	0.599087	0.638147

graphs, the time complexity of Sahni-Gonzalez algorithm is $O(|V|^2)$.

Next, we present the results obtained by the edge contraction heuristic (C), the Sahni-Gonzalez algorithm (SG), the compromise heuristic (CSG) and the modified versions, $SG1$, $SG2$ and $SG3$ of SG in Table 6. Here the results are presented as the ratios of the cut value of the solution obtained using a given algorithm to the total edge weight W. We see that the compromise heuristic (CSG) does not perform better than the contraction and Sahni-Gonzalez algorithms. But the results we obtained from the algorithms $SG1$, $SG2$ and $SG3$ are more encouraging. Among these $SG3$ is the best overall. As it was explained in the previous section, $SG3$ chooses the "best" candidate at each iteration by considering the absolute contribution in terms of both the increase in the cut value and the increase in the partition weight. Figure 26 illustrates the results graphically for the SG, C and $SG3$ algorithms. It clearly shows that $SG3$ outperforms the contraction and Sahni-Gonzalez algorithms in all considered instances.

Our experimental analysis shows that $SG3$ is the best choice although it has the same worst-case approximation ratio of 1/2 as the original Sahni-Gonzalez algorithm. It is important to note that experimental analysis has a crucial role especially in comparison of construction algorithms for max-cut since theoretically it is not possible to obtain an approximation ratio greater than 1/2 as long as the total edge weight is used instead of the optimal cut value in the approximation ratio derivation.

VII.6. Conclusion

In this chapter, a greedy worst-out construction heuristic for the max-cut problem called the edge contraction heuristic was introduced. We have shown that it has an approximation ratio of at least 1/3 and a time complexity of $O(|V|^3)$. To the best of

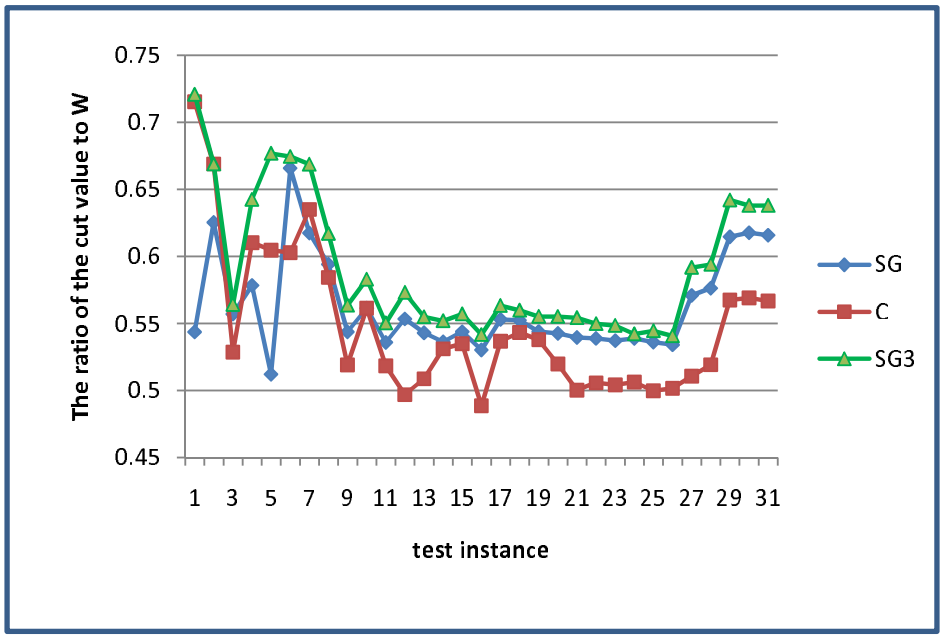

Fig. 26 Comparison of results for SG, C, and $SG3$ algorithms. The instances are numbered in the order they are presented in Tables 5 and 6.

our knowledge, this is the first time a worst-out greedy approach has been applied to approximate the max-cut problem. We also proposed several best-in $\frac{1}{2}$-approximation algorithms for the same problem that are modifications of a well-known heuristic introduced by Sahni and Gonzalez [70]. We carried out some numerical experiments to compare the performance of these heuristics. Our experiments showed that the edge contraction heuristic is outperformed by Sahni and Gonzalez approach. Moreover, we observed that its approximation ratio is worse than that of the Sahni-Gonzalez algorithm. We also observed that the modified versions of Sahni-Gonzalez heuristics, where a score function is used to determine the best candidate at each iteration, outperform the Sahni-Gonzalez heuristic. Based on our experiments, we concluded

that this approach is outperformed by best-in algorithms, while the results obtained from modified versions of the Sahni-Gonzalez approach are quite encouraging. Recall that these results are for construction algorithms only and can be further improved by applying improvement heuristics, such as local search and advanced metaheuristic strategies [1, 40].

CHAPTER VIII

CONCLUSION AND FUTURE WORK

This dissertation focuses on several related optimization problems that arise in geometric graphs. In particular, we investigate the computational complexity and approximability properties of several optimization problems in unit disk and unit ball graphs, and develop algorithms to find exact and approximate solutions. Furthermore, we establish complexity-based theoretical justification for several greedy heuristics. As outlined in Chapter I, this research comprises a series of linked objectives:

1. Survey optimization problems in unit disk graphs and work on extensions of the existing algorithms to unit ball graphs;

2. Identify open problems in unit disk and unit ball graphs regarding computational complexity and approximation status and propose solutions;

3. Develop and implement an exact solution procedure for the maximum *k-clique problem* in unit disk graphs;

4. Analyze and develop a centralized approximation algorithm for the *minimum k-bottleneck connected dominating set problem* in unit disk and unit ball graphs;

5. Establish complexity-based techniques for analysis of heuristics to provide a theoretical justification for the choice of construction heuristics;

6. Perform experimental comparison of several heuristics for the max-cut problem in general graphs.

Motivated by the wireless network applications, our initial goal was to identify the complexity and approximability of several optimization problems in unit disk

and unit ball graphs. In order to efficiently operate wireless networks, several optimization problems are employed. For instance, in order to prevent interference of transmission signals we need to assign different frequencies to the nodes whose transmission ranges intersect. This corresponds to a minimum vertex coloring problem in a graph theoretic representation of the network. The unit disk graph model is a simple graph theoretic model for wireless networks. This model assumes that wireless nodes are placed on the plane and each node has a unit transmission range that is omnidirectional. Two wireless nodes can communicate if they are within each other's transmission region. The unit disk model has been widely studied in the literature. Many NP-hard problems preserve their computational complexity even when restricted to unit disk graphs. However, the structure of these graphs enable efficient approximation algorithms. Although a three-dimensional representation seems to be more realistic for these networks, e.g., in ocean monitoring, there is not much research published in this area. Thus, our initial goal was to do a literature review on optimization problems in unit disk graphs and investigate if they are applicable when the graph is three dimensional, i.e., a unit ball graph. Chapter III is devoted to this analysis. In this chapter, we determined important geometrical properties for both unit disk and unit ball graphs. These properties are important in establishing complexity and approximability results for several optimization problems. Next, we presented extensions of several approximation algorithms for unit disk graphs to unit ball graphs. We observed that in most cases, the extension is straightforward. We outlined the steps of several approximation algorithms for UBGs and provided the corresponding performance guarantees. We considered domination, minimum vertex coloring, maximum independent set, max-cut, max-bisection, min-bisection, vertex cover, maximum clique and minimum clique partition problems. Although the unit disk graph models have been studied for many years, complexity or approximation

status of several optimization problems such as min-bisection and minimum clique partition are still open problems in unit disk graphs. Chapter III also presents a list of interesting open problems identified during the literature survey. These are:

1. The computational complexity and approximation status of min-bisection problem in unit disk and unit ball graphs;

2. The computational complexity and the existence of a PTAS for the maximum clique problem in unit ball graphs;

3. Existence of a PTAS for minimum clique partition problem on both unit disk and unit ball graphs.

We proved that, when the input graph is vertex weighted, then the minimum bisection problem is NP-hard in unit disk graphs. However the unweighted version is not proven yet. Furthermore, to our best knowledge, there are not any known approximation algorithms for this problem in unit disk graphs. The maximum clique problem is solvable in polynomial time in unit disk graphs [29]. Although there is a proof of NP-hardness by Afshani and Hatami [3] for some higher dimension for the same problem, the complexity in unit ball graphs is unknown. On the approximation side, the best known approximation ratio is 2.553. The minimum clique partition problem is NP-hard in unit disk and unit ball graphs. There is a 3-approximation algorithm for unit disk graphs, which can be extended to a 20-approximation algorithm for unit ball graphs. Thus, it is an interesting research question whether it is possible to further improve these ratios, ideally whether it is possible to have a PTAS or not. We believe that all these problems listed above are interesting research problems that will contribute to the field of graph theory and applications. These problems pose interesting future research directions.

During our literature survey presented in Chapter III, we identified that the maximum k-clique problem has not been studied in the context of unit disk graphs. This problem is interesting also because a related famous problem, the maximum clique, is solvable in polynomial time in unit disk graphs. Thus, one of our research goals was to analyze this problem and propose efficient solution procedures. This research objective is addressed in Chapter IV. Even though it has not been established formally, we have strong indications that this problem is NP-hard in unit disk graphs. In particular, we presented complexity results of clique problems in other geometric graphs that may be helpful in the complexity analysis. We presented a maximal by inclusion greedy heuristic for general graphs. Using the idea behind the polynomial algorithm for the maximum clique problem, we developed a matching-based branch and bound algorithm for the exact solution in unit disk graphs. The main idea is that a k-clique corresponds to a clique on the k^{th} power of the input graph. The k^{th} power of a unit disk graph is not necessarily a unit disk graph but it is a subgraph of another unit disk graph whose unit distance is defined as k times the original unit distance. In a unit disk graph, the maximum clique is contained in a special subgraph of the intersection of two adjacent vertices. This special subgraph is a co-bipartite graph. Thus, using matching in the complement of each subgraph, we can find the maximum clique. For the maximum k-clique, we can employ the same technique in a branch and bound scheme. Contrary to the traditional branch and bound algorithms which rely on the linear relaxations, our method relies on the existence of additional edges, which are named as "fake", that makes the problem solvable by using a matching algorithm on the complement. Our branching strategy is determined by the existence of "fake" edges in the solution compared to the traditional integrality constraints. We presented the details of our algorithm and implementation. Furthermore, we demonstrated the effectiveness of our algorithm through computational experiments.

For comparison purposes, we used the 1-plex formulation on the k^{th} power of the input graph and solved our test instances using the default CPLEX solver. In very dense instances, where edge density is around 0.9, 1-plex formulation outperformed our algorithm. However in sparser instances we observed that our method is much more efficient. In fact, if we consider wireless network applications, it is more likely to have sparser instances since denser instances will yield more signal interference and also require more energy. When we ran our algorithms for graphs with 1000 vertices with an edge density around 0.2, we observed that 1-plex formulation failed to give the optimal solution in a reasonable time. So we introduced time limits for CPLEX. On the other hand, we can easily get the optimal solution by our matching based branch and bound algorithm. On the average, our algorithm found the optimal solution in 98 seconds whereas on an average of 3164 seconds, the 1-plex formulation only showed that the solution output had a gap which was more than 75%. As a future work, our first goal is to prove the computational complexity of this problem in unit disk and unit ball graphs. We believe that further improvement of our solution procedure is possible. In addition, we would like to design approximation algorithms for both unit disk and unit ball graphs, as well as an exact solution procedure for unit ball graphs. Note that the current approach cannot be extended to unit ball graphs since they lack the crucial property. But, using geometrical information can provide effective solution procedures for unit ball graphs as well. We believe that by some modifications our approach can be used to solve the maximum k-club in unit disk graphs as well. We propose the analysis of computational complexity and approximability of clique relaxations such as k-plex and k-club as a future research goal.

In Chapter V, we investigated the minimum connected bottleneck connected dominating set problem in unit ball graphs. We proposed this problem as a viable

approach to find the optimal transmission range of a wireless network with respect to a certain size of "virtual backbone", which is a (small) subset of nodes that are used as a core for communication within the network. In traditional models it is assumed that the range of communication or its estimate is given in advance and is used as an input. In reality, the choice of the transmission range is an important decision variable that effects all the operations of the network. The energy usage of a wireless node is directly related to its transmission range. Thus, we want to minimize energy usage by minimizing the transmission range. At the same time, we want to make sure that we have a certain size of virtual backbone. We presented a literature review on transmission range determination. Most of the techniques do not focus on a unit disk graph model. We found out that the goal of ensuring a certain size of "virtual backbone" has not been studied before. Since virtual backbone corresponds to a connected dominating set in a graph representation, we proposed using minimum k-BCDS problem, which is the minimum bottleneck cost problem that yields a connected dominating set of size k. We observed that the bottleneck dominating set problems studied in graph theory focus on vertex weighted cases and are generally solvable in polynomial time. We proved that the minimum k-BCDS problem is NP-hard and not approximable within a factor of $2-\epsilon$, even when restricted to graphs whose edge weights satisfy triangle inequality. We proposed a 3-approximation algorithm for this special case and further proved that by iteratively running a PTAS algorithm with $\epsilon = 1/(k+1)$ we can solve this problem in polynomial time in unit disk and unit ball graphs. The algorithm requires at most $O(n^2)$ steps where each step takes $O(n^{2+(k+1)^3})$ time. Our future goal is to find more efficient solution procedures for this problem in unit disk and unit ball graphs.

In Chapter VI, we represented our results on theoretical justifications of simple greedy heuristics. Our motivation for this research was the lack of theoretical foun-

dations behind simple greedy heuristics in spite of the fact that they are widely used in solving large scale NP-hard problems. Especially for optimization problems for which it is NP-hard to approximate within a factor of n^{ϵ}, e.g. maximum clique, the approximation ratio of the heuristic can not be used as a claim for being the best. Moreover many simple heuristics also lack an approximation ratio. Thus, our goal is to establish complexity based techniques that will help to characterize the choice of simple greedy heuristics. An analogue of this problem is well known for best approximation algorithms. For instance, the k-center problem is not approximable within a factor of $2 - \epsilon$ for any $\epsilon > 0$ and any approximation algorithm with a performance ratio of 2 is considered to be the "best" possible. We first proved that the gap between k-club and l-club numbers is NP-hard to recognize for any $k > l$. Using this, we proved that it is not possible to have a polynomial time approximation algorithm that will always guarantee a solution of size $\Delta + 1$, where Δ is the maximum vertex degree. Motivated by this result, we investigated the maximum independent set, maximum clique, maximum k-plex, minimum vertex color and minimum clique partition problems. For the minimum vertex coloring problem, we outlined a heuristic that always outputs a Δ-coloring whenever there exists a vertex in the input graph which has a degree less than Δ. Based on the fact that 3-colorability of graphs with maximum degree 4 is NP-complete, we concluded that the gap recognition problem between the chromatic number and Δ is NP-hard and thus the Δ-coloring heuristic that we outlined is "provably best". Since minimum clique partition is equivalent to minimum vertex coloring on the complement of the input graph, this result also applies to it. For the maximum independent set (maximum clique) and the maximum k-plex problems, we proved that a simple maximal by inclusion heuristic is the "best" by showing that it is NP-hard to recognize the gap between the heuristic solution and the optimal solution values. Thus, existence of any polynomial-time

algorithm which always guarantees a better solution than the heuristic (whenever a better solution exists) is impossible, unless $P = NP$. Finally, we conjectured that the same technique can be applied to a broader class of graph problems that also includes maximum clique, maximum k-plex and maximum independent set problems. The *maximum subgraph with property π problem* is finding a subgraph of maximum cardinality that satisfies some given property π. We focused on properties that are hereditary on induced subgraphs, additive or co-additive, nontrivial and interesting. For such properties, the *maximum subgraph with property π problem* is NP-hard. We presented some arguments that can be used in proving that any maximal by inclusion greedy heuristic based on some order of vertices is the "best" in the sense that it is not possible to have a polynomial time approximation algorithm which always gives a better solution. This class includes several optimization problems including maximum planar subgraph, maximum bipartite subgraph, maximum clique, maximum outerplanar subgraph and maximum degree-constrained subgraph problems. The results that we obtained can also be viewed as an additional evidence of these problems' practical intractability. On the other hand, this should not prevent the practitioners from designing more sophisticated approaches for these problems, since the above result describes just the worst-case behavior of the heuristics. Our immediate future goal is to prove our conjecture.

Finally, Chapter VII demonstrates our research findings on the construction heuristics for the max-cut problem. We proposed a "worst-out" approach based on contracting edges of minimum weight iteratively. We have shown that it has an approximation ratio of at least $1/3$ and a time complexity of $O(|V|^3)$. To the best of our knowledge, this is the first time a worst-out greedy approach has been applied to approximate the max-cut problem. We also proposed several "best-in" $\frac{1}{2}$-approximation algorithms for the same problem that are modifications of a well

known heuristic introduced by Sahni and Gonzalez [70]. Based on our experiments, we concluded that the "worst-out" approach is outperformed by best-in algorithms, while the results obtained from modified versions of the Sahni-Gonzalez approach are quite encouraging.

REFERENCES

1. Aarts, E., Lenstra, J. (eds.): Local search in combinatorial optimization. Princeton University Press, Princeton, NJ (2003)

2. Afshani, P., Chan, T.: Approximation algorithms for maximum cliques in 3d unit-disk graphs. In: Canadian Conference on Computational Geometry, pp. 19–22. http://www.cccg.ca/proceedings/2005/69.pdf (2005)

3. Afshani, P., Hatami, H.: Approximation and inapproximability results for maximum clique of disc graphs in high dimensions. Information Processing Letters **105**, 83–87 (2008)

4. Agrawal, R., Rajagopalan, S., Srikant, R., Xu, Y.: Mining newsgroups using networks arising from social behavior. In: Proceedings of the Twelfth International World Wide Web Conference, pp. 529–535 (2003)

5. Akyildiz, I.F., Pompili, D., Melodia, T.: Underwater acoustic sensor networks: research challenges. Ad Hoc Networks **3**, 257–279 (2005)

6. Alon, N., Feige, U., Wigderson, A., Zuckerman, D.: Derandomized graph products. Computational Complexity **5**, 60–75 (1995)

7. Alperin, H., Nowak, I.: Lagrangian smoothing heuristics for max-cut. Journal of Heuristics **11**, 447–463 (2005)

8. Ambuhl, C., Wagner, U.: On the clique problem in intersection graphs of ellipses. Lecture Notes in Computer Science **2518**, 141–155 (2002)

9. Arora, S., Karger, D., Karpinski, M.: Polynomial time approximation schemes for dense instances of NP-hard problems. Journal of Computer and System Sciences **58**, 193–210 (1999)

10. Balasundaram, B.: Graph theoretic generalization of clique: optimization and extensions. Ph.D. thesis, Texas A&M University, College Station (2007)

11. Balasundaram, B., Butenko., S.: Constructing test functions for global optimization using continuous formulations of optimization problems on graphs. Optimization Methods and Software **20**, 1–14 (2005)

12. Balasundaram, B., Butenko, S.: Optimization problems in unit-disk graphs. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, 2nd edn., pp. 2832–2844. Spinger Science + Business Media, New York (2008)

13. Balasundaram, B., Butenko, S., Trukhanov, S.: Novel approaches for analyzing biological networks. Journal of Combinatorial Optimization **10**, 23–39 (2005)

14. Bar-Yehuda, R., Even, S.: A local ratio theorem for approximating the weighted vertex cover problem. Analysis and Design of Algorithms for Combinatorial Problems, Annals of Discrete Mathematics **25**, 27–45 (1985)

15. Barahona, F.: The max-cut problem in graphs not contractible to K_5. Operations Research Letters **2**, 107–111 (1983)

16. Barahona, F., Grotschel, M., Junger, M., Reinelt, G.: An application of combinatorial optimization to statistical physics and circuit layout design. Operations Research **36**, 493–513 (1988)

17. Bramoullé, Y.: Anti-coordination and social interactions. Games and Economic Behavior **58**(1), 30–49 (2007)

18. Breu, H., Kirkpatrick, D.G.: Unit disk graph recognition is NP-hard. Computational Geometry. Theory and Applications **9**, 9–31 (1998)

19. Burer, S., Monteiro, R.C., Zhang, Y.: Rank-two relaxation heuristics for maxcut and other binary quadratic programs. SIAM Journal on Optimization **12**, 503–521 (2001)

20. Burer, S., Monteiro, R.D.C.: A projected gradient algorithm for solving the maxcut SDP relaxation. Optimization Methods and Software **15**, 175–200 (2001)

21. Busygin, S., Pasechnik, D.: On NP-hardness of the clique partition – independence number gap recognition and related problems. Discrete Mathematics **304**, 460–463 (2006)

22. Butman, A., Hermelin, D., Lewenstein, M., Rawitz, D.: Optimization problems in multiple-interval graphs. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 268–277 (2007)

23. Cerioli, M.R., Faria, L., Ferreira, T.O., Protti, F.: On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation. Electronic Notes in Discrete Mathematics **18**, 73–79 (2004)

24. Ceroi, S.: The clique number of unit quasi-disk graphs. Rapport de recherche 4419, INRIA (2002). URL http://hal.inria.fr/inria-00072169/en/

25. Chen, Y., Crawford, M., Ghosh, J.: Integrating support vector machines in a hierarchical output decomposition framework. In: Proceedings of the International Geoscience and Remote Sensing Symposium, pp. 949–953 (2004)

26. Cheng, X., Huang, X., Li, D., Wu, W., Du, D.: A polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks. Networks **42**, 202–208 (2003)

27. Cho, J.D., Raje, S., Sarrafzadeh, M.: Fast approximation algorithms on maxcut, k-coloring, and k-color ordering for VLSI applications. IEEE Transactions on Computers **47**(11), 1253–1266 (1998)

28. Chudnovsky, M., Cornuéjols, G., Liu, X., Seymour, P., Vuskovic, K.: Recognizing berge graphs. Combinatorica **25**, 143–186 (2005)

29. Clark, B.N., Coulbourn, C.J., Johnson, D.S.: Unit disk graphs. Discrete Mathematics **86**, 165–177 (1990)

30. Deng, J., Han, Y.S., Chen, P., Varshney, P.K.: Optimal transmission range for wireless ad hoc networks based on energy efficiency. IEEE Transactions on Communications **55**(9), 1772–1781 (2007)

31. Deza, M., Laurent, M.: Geometry of cuts and metrics. Springer Verlag, Berlin Heidelberg, Germany (1997)

32. Díaz, J., Kamiński, M.: Max-cut and max-bisection are NP-hard on unit disk graphs. Theoretical Computer Science **377**, 271–276 (2007)

33. Díaz, J., Penrose, M.D., Petit, J., Serna, M.: Approximating layout problems on random geometric graphs. Journal of Algorithms **39**(1), 78–116 (2001)

34. Durocher, S., Kirkpatrick, D., Narayanan, L.: On routing with guaranteed delivery in three-dimensional ad hoc wireless networks. Lecture Notes in Computer Science **4904**, 546–557 (2008)

35. Edachery, J., Sen, A., Brandenburg, F.J.: Graph clustering using distance-k cliques. Lecture Notes in Computer Science **1731**, 98–106 (1999)

36. Erlebach, T., Fiala, J.: Independence and coloring problems on intersection graphs of disks. Lecture Notes in Computer Science **3484**, 135–155 (2006)

37. Festa, P., Pardalos, P.M., Resende, M.G.C., Ribeiro, C.C.: Randomized heuristics for the max-cut problem. Optimization Methods and Software **7**, 1033–1058 (2002)

38. Funke, S., Kesselman, A., Meyer, U., Segal, M.: A simple improved distributed algorithm for minimum cds in unit disk graphs. ACM Transactions on Sensor Networks **2**(3), 444–453 (2006)

39. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company, New York (1979)

40. Glover, F., Kochenberger, G.: Handbook of metaheuristics. Kluwer Academic Publishers, Norwell, Massachusetts (2003)

41. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the Association for Computing Machinery **42**(6), 1115–1145 (1995)

42. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization. 2nd edn. Springer-Verlag, Berlin (1993)

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

44. Hadlock, F.O.: Finding a maximum cut of a planar graph in polynomial time. SIAM Journal on Computing **4**, 221–225 (1975)

45. Haglin, D.J., Venkatesan, S.M.: Approximation and intractability results for the maximum cut problem and its variants. IEEE Transactions on Computers **40**, 110–113 (1991)

46. Hales, T.C.: Cannonballs and honeycombs. Notices of the AMS **47**(4), 440–449 (2000)

47. Halldórsson, M.M.: Approximating discrete collections via local improvements. In: Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 160–169 (1995)

48. Halperin, E., Livnat, D., Zwick, U.: Max-cut in cubic graphs. In: Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 506–513 (2002)

49. Hastad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Mathematica **182**, 105–142 (1999)

50. Hliněný, P.: Touching graphs of unit balls. Lecture Notes in Computer Science **1353**, 350–358 (1997)

51. Hochbaum, D.S.: Efficient bounds for the stable set, vertex cover and set packing problems. Discrete Applied Mathematics **6**, 243–254 (1983)

52. Hochbaum, D.S., Shmoys, D.B.: A unified approach to approximation algorithms for bottleneck problems. Journal of the Association for Computing Machinery **33**, 533–550 (1986)

53. Huang, Y., Gao, X., Zhang, Z., Wu, W.: A better constant-factor approximation for weighted dominating set in unit disk graph. Journal of Combinatorial Optimization **18**(2), 179–194 (2009)

54. Hunt, H.B., Marathe, M.V., Radhakrishnan, V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. Journal of Algorithms **26**(2), 238–274 (1998)

55. Jansen, K., Karpinski, M., Lingas, A., Seidel, E.: Polynomial time approximation schemes for max-bisection on planar and geometric graphs. SIAM Journal on Computing **35**(1), 110–119 (2005)

56. Kajitani, Y., Cho, J.D., Sarrafzadeh, M.: New approximation results on graph matching and related problems. Lecture Notes in Computer Science **903**, 343–358 (1994)

57. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)

58. Karpinski, M., Feige, U., Landberg, M.: Improved approximation of max-cut on graphs of bounded degree. Journal of Algorithms **43**, 201–219 (2002)

59. Kloks, T., Kratsch, D., Lee, C.M., Liu, J.: Improved bottleneck domination algorithms. Discrete Applied Mathematics **154**, 1578–1592 (2006)

60. Kratochvil, J., Kubena, A.: On intersection representations of co-planar graphs. Discrete Mathematics **178**, 251–255 (1998)

61. Lewis, J., Yannakakis, M.: The node deletion problem for hereditary properties is NP-complete. Journal of Computer and System Sciences **20**, 219–230 (1980)

62. Li, L., Halpern, J.Y.: Minimum-energy mobile wireless networks revisited. In: Proceedings of the IEEE International Conference on Communications, pp. 278–283 (2001)

63. Li, X.Y., Stojmenovic, I.: Broadcasting and topology control in wireless ad hoc networks. In: Boukerche, A. (ed.) Handbook of Algorithms for Wireless Networking and Mobile Computing, pp. 239–261. CRC, Boca Raton, Florida (2006)

64. Liu, H., Liu, S., Xu, F.: A tight semidefnite relaxation of the max-cut problem. Journal of Combinatorial Optimization **7**, 237–245 (2003)

65. Maifray, F., Preissmann, M.: On the NP-completeness of the k-colourability problem for triangle-free graphs. Discrete Mathematics **162**, 313–317 (1996)

66. Marathe, M.V., Breu, H., Hunt, H.B., Ravi, S.S., Rosenkrantz, D.J.: Simple heuristics for unit disk graphs. Networks **25**, 59–68 (1995)

67. Matsui, T.: Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs **1763**, 194–200 (2004)

68. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. Journal of Computational System Science **43**, 425–440 (1991)

69. Rodoplu, V., Meng, T.H.: Minimum energy mobile wireless networks. IEEE Journal on Selected Areas in Communications **17**(8), 1333–1344 (1999)

70. Sahni, S., Gonzales, T.: P-complete approximation problems. Journal of the Associaton for Computing Machinery **23**(3), 555–565 (1976)

71. Sanchez, M., Manzoni, P., Haas, Z.J.: Determination of critical transmission range in ad hoc networks. In: Proceedings of Multiaccess, Mobility and Teletraffic for Wireless Communications 1999 Workshop, pp. 6–8 (1999)

72. Schmid, S., Wattenhofer, R.: Algorithmic models for sensor networks. In: Parallel and Distributed Processing Symposium, 2006. 20th International, pp. 11–21 (2006)

73. Shang, W., Yao, F., Wan, P., Hu, X.: On minimum m-connected k-dominating set problem in unit disc graphs. Journal of Combinatorial Optimization **16**, 99–106 (2008)

74. Szekeres, G., Wilf, H.: An inequality for the chromatic number of a graph. Journal of Combinatorial Theory, Series B **4**, 1–3 (1968)

75. T. Erlebach, K.J., Seidel, E.: Polynomial time approximation schemes for geometric intersection graphs. SIAM Journal on Computing **34**(6), 1302–1323 (2005)

76. Tai, M.T., Zhang, N., Tiwari, R., Xu, X.: On approximation algrithms of k-connected m-dominating sets in disk graphs. Theoretical Computer Science **385**, 49–59 (2007)

77. Trevisan, L., Sorkin, G.B., Sudan, M., Williamson, D.P.: Gadgets, approximation, and linear programming. SIAM Journal on Computing **29**(6), 2074–2097 (2000)

78. de la Vega, W.F.: Max-cut has a randomized approximation scheme in dense graphs. Random Structures and Algorithms **8**(3), 187–198 (1996)

79. de la Vega, W.F., Kenyon, C.: A randomized approximation scheme for metric max-cut. In: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, pp. 468–475 (1996)

80. Wu, B., Huang, L., Lin, C.: On the minimum density of 2-clique and 3/2-clique in directed graphs. http://algo2008.csie.chu.edu.tw/file/f37.pdf. Accessed 15 May 2009

81. Wu, J., Li, H.: On calculating connected dominating set for efficient routing in ad hoc wireless networks. In: Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 7–14 (1999)

82. Wu, J., Wu, B.: A transmission range reduction scheme for power-aware broadcasting in ad hoc networks using connected dominating sets. IEEE-Vehicular Technology Conference **5**, 2906–2909 (2003)

83. Wu, W., Du, H., Jia, X., Li, Y., Huang, S.: Minimum connected dominating sets and maximal independent sets in unit disk graphs. Theoretical Computer Science **352**, 1–7 (2006)

84. Yen, W.C.K.: Bottleneck domination and bottleneck independent domination on graphs. Journal of Information Science and Engineering **18**, 311–331 (2002)

85. Zhang, Z., Gao, X., Wu, W., Du, D.: PTAS for minimum connected dominating set in unit ball graph. Lecture Notes in Computer Science **5258**, 154–161 (2008)

VITA

Sera Kahruman-Anderoglu received her Bachelor of Science degree in industrial engineering from Bogazici University, Istanbul, Turkey in 2003. In August 2003, she started the Ph.D. program at the Industrial and Systems Engineering Department of Texas A&M University in College Station. She received her Ph.D. degree in December 2009. During her doctoral studies, she held positions as research assistant, teaching assistant and lecturer. Her research interests are in combinatorial optimization, graph theory, operations research, healthcare and wireless network applications of operations research and stochastic optimization.

Permanent address:

Cinar Mh.
10/1 Sk. No: 17/2
Bagcilar/Istanbul 34200
TURKEY

This document was typeset in LaTeX by the author.