Approximating Node-Weighted Steiner Subgraphs for Multicast Communication in Wireless Networks

Thesis submitted for the degree of

Doctor of Philosophy

at the University of Leicester

by

Ambreen Shahnaz

Department of Computer Science

University of Leicester

September 2011

Declaration

This submission is my own work done under supervision of Prof. Thomas Erlebach in the Department of Computer Science, University of Leicester. To the best of my knowledge, the material of this submission has not been previously published for the award of any other degree or diploma at any other university or institute. The contents submitted are the result of my own research except where due acknowledgement has been made. Chapter 5 and Chapter 6 are based on publications [15] and [50], respectively.

Approximating Node-Weighted Steiner Subgraphs for Multicast Communication in Wireless Networks

Ambreen Shahnaz

Abstract

We are motivated by the problem of computing multicast routing structures in wireless ad-hoc networks modelled by special classes of graphs including unit disk graphs, quasi-unit disk graphs and $(\lambda + 1)$-claw-free graphs. Multicast communication can be established by a tree known as Steiner tree. Wireless ad-hoc networks must operate using limited resources, therefore, the suitability of nodes for inclusion in a Steiner tree can vary widely between different nodes. We model this by assuming that each node of the network is assigned a weight that represents the cost of including it in the Steiner tree. Our goal is to compute a Steiner tree with minimum total node weight. However, in scenarios where nodes and links are not reliable, a tree has the drawback that it can be disconnected by the failure of even a single link or node in the network. Therefore, we also consider various fault-tolerant routing structures called 2-edge-connected Steiner subgraphs, k-edge-connected Steiner subgraphs, 2-vertex-connected Steiner subgraphs, and 2-edge-connected group Steiner subgraphs. The problems we consider are NP-hard, so we are interested in algorithms that compute provably good approximate solutions in polynomial time. We present a generalization of Steiner subgraph problems referred to as the node-weighted δ-Steiner subgraph problem, where δ represents connectivity requirements. We present an algorithm with approximation ratio $0.5d\rho$ for the node-weighted δ-Steiner subgraph problem, where d is the bounded maximum degree of the solution subgraph, and ρ is the approximation ratio of the edge-weighted version of the δ-Steiner subgraph problem. We then shown how to construct solution subgraphs of bounded maximum degree d in several graph classes for our problem variants. As a result, we obtain algorithms for the problems we consider, on graph classes that admit subgraphs of small degree, whose approximation ratios are better than the best known ratios for the same problems on general graphs.

Acknowledgements

I express my immense gratitude to Almighty who blessed me in every way. I am deeply indebted to my supervisor Prof. Thomas Erlebach who has supported me throughout my thesis with his patience and knowledge. The help and advice I received from Thomas is invaluable. I have learnt and benefited a lot from his expertise. Thank you Thomas for the support and guidance, and for finding time for me despite busy schedules and many commitments whenever I needed to discuss any issue.

I would like to thank my co-supervisor, Dr. Stanley P.Y. Fung, and Dr. Fer-Jan de Vries for being on the thesis committee and providing feedback. I am grateful to all faculty and staff members of the department and the university for providing a good working environment. I thank the department for arranging useful seminars and PhD short courses. I would like to acknowledge the academic and technical facilities of the university.

It is my pleasure to acknowledge Prof. M. A. Irfan and Dr. M. Waqar Ali Asad for their encouragement and guidance before I started PhD studies. Many thanks go in particular to Frontier Women University Peshawar and Higher Education Commission Pakistan for funding my studies here.

My special thanks are reserved for my dear parents for their love and encouragement during all stages of my studies and my life, and thanks to my siblings Amna and Azam for their love and support. I am lucky that I found very good friends here, Nosheen, Afia and Nadia, whose companionship I enjoyed a lot. Thanks to my colleagues for the nice time we spent in the department, and to friends with whom I had enjoyable tea-breaks. Last but not least, I would like to thank Iftikhar Ahmad for useful discussions on algorithms and for suggestions during thesis writing.

Ambreen Shahnaz
Leicester

Contents

Chapter 1

Introduction

A wireless ad-hoc network is a multi-hop network without any fixed infrastructure. Nodes of the network are mobile or static and communicate through radio waves by relaying packets in the absence of central management. This makes wireless ad-hoc networks different from cellular networks where base stations are installed to manage routing centrally. An important application of wireless ad-hoc networks is in situations like disaster and recovery, when there is no infrastructure of a communication network or the existing network has collapsed, and the deployment of a network is highly required. Other applications are in environmental monitoring, military operations, and video conferencing among a group of people. There are several types of these networks including wireless sensor networks, wireless mesh networks and vehicular wireless ad-hoc networks [40, 54].

The amount of available resources, for example, battery power, memory, and bandwidth, is very limited in wireless ad-hoc networks. Therefore, due to the absence of any fixed infrastructure, and limited resources these networks pose numerous challenges. To overcome the challenges and ensure efficient use of limited resources, these networks have been studied in different domains such as energy-efficiency, routing, and fault-tolerance. A simple approach to the study of wireless

ad-hoc networks is to model them as unit disk graphs. A unit disk graph is a graph whose nodes correspond to equisized disks in the plane with an edge between two nodes if the corresponding disks intersect [38]. Such a graph is a simplified model of a wireless network consisting of nodes with omnidirectional antennas and equal transmission power. Nodes can communicate directly when they are in each other's transmission range. Since the unit disk graph model is too idealized, more general graph models that are a better reflection of real wireless ad-hoc networks have been proposed. Kuhn et al. [35] employ *quasi-unit disk graphs* or α*-unit disk graphs*, where nodes are adjacent if their distance is at most α, non-adjacent if their distance is greater than 1, and can be adjacent or non-adjacent otherwise. There is another graph class considered in [34, 33, 49] called the class of *bounded independence graphs*. Let the r-neighborhood of a node u be the set of nodes that are at most r hops away from u. Bounded independence graphs are the graphs in which the maximum number of independent nodes in the r-neighborhood of any node is bounded by a polynomial $p(r)$ [49]. Another class of graphs is the class of *bi-directional disk graphs* in which disks can be of different size [53]. For two nodes u and v in a bi-directional disk graph, there is an edge u-v if both u and v can hear each other's transmissions directly. We consider the class of $(\lambda+1)$*-claw-free graphs* which is more general and includes unit-disk graphs, α-unit disk graphs, bounded independence graphs, and bi-directional disk graphs. A graph is $(\lambda+1)$-claw-free if every node has at most λ independent neighbors. Since the claw-freeness property is exhibited by many other classes of graphs considered in this thesis, the results we achieve for $(\lambda + 1)$-claw-free graphs are applicable to those classes as well. We only concentrate on static ad-hoc networks modelled by the above classes of graphs and do not deal with mobility issues.

We are interested in multicast communication in which a message is aimed to be delivered to a set of destinations from a source node in a network. Consider an

ad-hoc wireless network. Every node has some transmission range, and we say that two nodes are adjacent if they can transmit a message to each other directly. Let a subset of the nodes of the network want to initiate a multicast session. The idea is to model this network using a graph and connect all members of this subset by a tree. To communicate with distant nodes, multi-hop transmissions are required for which intermediate nodes are necessary. Thus, apart from source and destinations, there can be several other nodes that participate in the communication only to forward messages to the destination nodes. This type of communication can be modelled as a multicast tree known as Steiner tree. The multicast sender and receivers correspond to terminals, and other nodes participating in the tree are Steiner nodes. Given a graph $G = (V, E)$, $K \subseteq V$, a tree T that is a subgraph of G is a Steiner tree if it spans all vertices in K. As Steiner nodes are nodes that participate in the multicast tree by forwarding packets but do not benefit from the multicast, it is a natural objective to compute a tree that minimizes the total cost of the Steiner nodes. One can therefore consider the node-weighted version of the Steiner tree problem defined as follows: Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for $v \in V$ and a subset of nodes $K \subseteq V$ (*terminals*), compute a subgraph T of G that is a tree and contains all the nodes in K [62]. The objective is to minimize the total weight of the vertices of T.

A disadvantage of multicast trees is that they are not reliable for communication because there is a single path between sender and receivers, and links may break because of various reasons like interference and collision of packets etc. There can also be node failures in the network due to some fault in the device or scarce energy resources. Therefore, a network needs to be robust to deal with such failures, i.e., routing structures are required in which sender and receivers are connected even if a link or node fails.

Unfortunately, there are hundreds of problems for which optimal solutions

cannot be computed efficiently despite years of effort. Such problems are classified as NP-hard problems. In this thesis, we consider the node-weighted Steiner tree problem which is NP-hard even in unit disk graphs [11]. We also consider node-weighted Steiner problems specified by certain higher connectivity requirements. We study the problems in several special classes of graphs that can be used to model wireless ad-hoc networks. We present approximation algorithms for each of the problem variants, and analyze their approximation ratio. The main ingredient of each algorithm's analysis is a proof showing that in special classes of graphs, there always exists a solution of bounded degree. This allows us to derive approximation results for each problem variant from a known approximation algorithm for the edge-weighted version. For graph classes that admit subgraphs of maximum degree bounded by d, we obtain a $0.5d\rho$-approximation algorithm for each node-weighted problem variant, where ρ is the approximation ratio of the algorithm for the edge-weighted version of the problem. Finding subgraphs of bounded maximum degree in the considered classes of graphs could also be of independent interest.

The problem variants and our results are briefly described below.

Overview of the Results

Node-Weighted δ-Steiner Subgraph Problem

We consider properties of the form $\delta(H, K)$, where H is a graph and K a set of vertices. Intuitively, the property $\delta(H, K)$ represents that H contains all vertices from K and satisfies certain connectivity requirements. In this thesis, we consider several special cases of such properties, for example, $\mu(H, K)$, $\sigma(H, K)$, $\varsigma(H, K)$, and $\psi_k(H, K)$, which will be described later.

For a fixed property δ, we define the *node-weighted δ-Steiner subgraph problem*

as follows: Given a graph $G = (V, E)$, a subset of vertices $K \subseteq V$, and nonnegative weights w_u for $u \in V$, the node-weighted δ-Steiner subgraph problem is to find a subgraph H of G such that the property $\delta(H, K)$ is satisfied. The objective of the node-weighted δ-Steiner subgraph problem is to minimize the sum of the weights of the vertex set of H. Since wireless nodes have limited resources, the weight w_u can represent the suitability of a node for inclusion in multicast communication. For example, if a node v has less remaining battery power than a node u, it means that v is less suitable for communication than u. Thus, v will have higher weight than u in the input graph, and the minimum-weight solution would prefer to include u as a Steiner node.

We present an algorithm with approximation ratio $0.5d\rho$ for the node-weighted δ-Steiner subgraph problem, where d is the maximum degree bound of the solution subgraph and ρ is the approximation ratio of the algorithm for the edge-weighted version. The node-weighted δ-Steiner subgraph problem generalizes several problem variants we study in this thesis.

Node-Weighted Minimum Steiner Tree Problem

We consider the property $\mu(H, K)$ which represents that H is a graph that contains all vertices in K and there is a path in H between every pair of vertices in K. The *node-weighted minimum Steiner tree problem* is defined as follows: Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for $v \in V$, and a subset of nodes $K \subseteq V$ called *terminals*, compute a subgraph H for G and K that satisfies the property $\mu(H, K)$. The objective is to minimize the total weight of the vertices of H. Note that the way in which we define the problem does not require the solution subgraph H to be a tree. However, it is easy to see that any solution subgraph can be transformed into a tree by removing redundant edges, without

increasing the weight of the solution.

For graph classes that admit spanning trees of maximum degree bounded by d, our algorithm yields approximation ratio $0.695d$ for the node-weighted minimum Steiner tree problem. We show the existence of spanning trees of bounded maximum degree in α-unit disk graphs and $(\lambda + 1)$-claw-free graphs. For unit disk graphs, a spanning tree of bounded degree was already known in the literature. For other graph classes considered in this thesis, the result of $(\lambda + 1)$-claw-free graphs is applied.

Node-Weighted 2-Edge-Connected Steiner Subgraph Problem

As trees are not reliable for communication, we therefore, consider Steiner subgraphs which are fault-tolerant to some extent. We consider the property $\sigma(H, K)$ representing that H is a graph that contains all vertices in K and has at least 2 edge-disjoint paths between every pair of vertices. We define the *node-weighted 2-edge-connected Steiner subgraph problem* ($NW2ECS$) as follows: Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for $v \in V$, and a subset of nodes $K \subseteq V$, compute a subgraph H for G and K that satisfies the property $\sigma(H, K)$. The goal of the node-weighted 2-edge-connected Steiner subgraph problem is to minimize the total weight of the vertices of H. Requiring only that H has 2 edge-disjoint paths between every pair of vertices in K does not change the problem.

We present an algorithm with approximation ratio d for the problem $NW2ECS$ in classes of graphs that admit 2-edge-connected spanning subgraphs of bounded maximum degree d for the solution subgraphs. We compute 2-edge-connected spanning subgraphs of bounded maximum degree d in unit-disk graphs, α-unit disk

graphs, and $(\lambda + 1)$-claw-free graphs.

Node-Weighted 2-Vertex-Connected Steiner Subgraph Problem

For the property of being resilient to node failures, we consider 2-vertex-connectivity. We consider the property $\varsigma(H, K)$ which expresses that H is a graph that contains all terminals and has at least 2 vertex disjoint paths between every pair of vertices. The *node-weighted 2-vertex-connected Steiner subgraph problem* ($NW2VCS$) is defined as follows: Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for all $v \in V$, and a set of terminals $K \subseteq V$, compute a subgraph H for G and K that satisfies the property $\varsigma(H, K)$. The objective of the problem $NW2VCS$ is to minimize the total weight of the vertices of H. The requirement of 2 vertex-disjoint paths only between every pair of terminals of H does not change the problem.

For a graph class that contains a 2-vertex-connected spanning subgraph of bounded maximum degree d for any 2-vertex-connected Steiner subgraph, we present a d-approximation algorithm for $NW2VCS$. We find a 2-vertex-connected spanning subgraph of bounded degree d in $(\lambda + 1)$-claw-free graphs.

Node-Weighted 2-Edge-Connected Group Steiner Subgraph Problem

We consider the property $\chi(H, K)$ which represents that H is a graph that contains at least 2 edge-disjoint paths from each group K_i of terminals to a root vertex r and the end-points in K_i of these two paths are distinct. Strictly speaking, $\chi(H, K)$ is not a special case of $\delta(H, K)$ because of the additional constraints satisfied in χ. Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for all $v \in V$, a root node $r \in V$, and a collection of M subsets of nodes $K = \{K_1, ..., K_M\}$ in

$V \setminus \{r\}$ with $|K_i| = 2$ for all i, compute a subgraph H of G that contains at least 2 edge-disjoint paths from each group K_i to r such that the end-points of these paths in K_i are distinct. The objective is to minimize the total cost of the vertices in H.

We show that there exists a 2-edge-connected group Steiner subgraph of bounded maximum degree d for any solution subgraph in the class of $(\lambda + 1)$-claw-free-graphs. We achieve approximation ratio $1.695d$ for the node-weighted 2-edge-connected group Steiner subgraph problem in $(\lambda + 1)$-claw-free graphs.

Node-Weighted k-Edge-Connected Steiner Subgraph Problem

We extend our study of higher edge-connectivity requirements to the node-weighted k-edge-connected Steiner subgraphs. We consider the property $\psi_k(H, K)$ which specifies that H is a graph that contains all vertices in K and has at least k edge-disjoint paths between every pair of terminals. The *node-weighted k-edge-connected Steiner subgraph* problem ($NWkECS$) is defined as follows: Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for $v \in V$, and a subset of nodes $K \subseteq V$, compute a subgraph H for G and K that satisfies the property $\psi_k(H, K)$. The goal of the node-weighted k-edge-connected Steiner subgraph problem is to minimize the total weight of the vertices of H.

In the classes of graphs that admit k-edge-connected Steiner subgraphs of maximum degree bounded by d, we present a d-approximation algorithm for the node-weighted k-edge-connected Steiner subgraph problem. We show that there exists a k-edge-connected Steiner subgraph of bounded maximum degree for a solution of $NWkECS$ in the class of $(\lambda + 1)$-claw-free graphs.

Steiner tree problem	$1.35 \ln \lvert K \rvert$ [25]
k-edge-connected Steiner subgraph problem	$k \cdot O(\ln \lvert K \rvert)$ [43]
2-vertex-connected Steiner subgraph problem	$O(\ln n)$ [43]
2-edge-connected group Steiner subgraph problem	$O(\ln n)$ [29]

Table 1.1: Best known results for node-weighted Steiner problems in general graphs. $|K|$ denotes the number of terminals.

	UDG	α-UDG	bi-directional disk graphs	$(\lambda+1)$-claw-free graphs
Steiner tree problem	3.475[1]	$0.695(4/\alpha^2 + 4/\alpha + 2)$	$(12.51 \lceil \log_2 m \rceil + 9.035)$	$0.695 \cdot (\lambda + 1)$
2-edge-connected Steiner subgraph problem	12	$2 \cdot (4/\alpha^2 + 4/\alpha + 2)$	$2 \cdot (18 \lceil \log_2 m \rceil + 13)$	$2(\lambda + 1)$
k-edge-connected Steiner subgraph problem	$(2^k - 1)5 + k$	$(2^k - 1)(4/\alpha^2 + 4/\alpha + 1) + k$	$(2^k - 1)(18 \lceil \log_2 m \rceil + 12) + k$	$(2^k - 1)\lambda + k$
2-vertex-connected Steiner subgraph problem	12	$2 \cdot (4/\alpha^2 + 4/\alpha + 2)$	$2 \cdot (18 \lceil \log_2 m \rceil + 13)$	$2(\lambda + 1)$
2-edge-connected group Steiner subgraph problem	20.34	$3.39 \cdot (4/\alpha^2 + 4/\alpha + 2)$	$3.39 \cdot (18 \lceil \log_2 m \rceil + 13)$	$3.39(\lambda + 1)$

Table 1.2: New results for node-weighted Steiner problems in special graph classes. m is the ratio between maximum and minimum transmission radius in a bi-directional disk graph.

Structure of the Thesis

Some basics about graphs, complexity theory and approximation algorithms are presented in Chapter 2. Chapter 3 contains a detailed literature review on Steiner trees and fault-tolerant Steiner subgraphs. In Chapter 4, we address the node-weighted δ-Steiner subgraph problem and provide a solution which is adapted for several variants of Steiner subgraph problems studied in subsequent chapters. The node-weighted minimum Steiner tree problem is studied in Chapter 5. In Chapter 6, we study the node-weighted 2-edge-connected Steiner subgraph problem, 2-edge-

[1] This result was achieved independently by Zou et al. [62]

connected group Steiner subgraph problem, and the 2-vertex-connected Steiner subgraph problem. In Chapter 7, we discuss the node-weighted k-edge-connected Steiner subgraph problem and also present a proof showing that the node-weighted k-edge-connected Steiner subgraph problem is NP-hard in unit-disk graphs. Conclusion and directions for future work are given in Chapter 8.

Chapter 5 on the node-weighted minimum Steiner tree problem and the part of Chapter 6 on the node-weighted 2-edge-connected Steiner subgraph problem are based on our papers [15] and [50] respectively.

The best known approximation results for node-weighted Steiner problems in general graphs are listed in Table 1.1. The new approximation results obtained in this thesis for special graph classes are shown in Table 1.2.

Chapter 2

Preliminaries

We present general background on graphs, computational complexity, and approximation algorithms. The terminology and concepts described in this chapter are used throughout this thesis.

2.1 Graphs

An *undirected graph* is represented by $G = (V, E)$ where V is the set of points of G called vertices, and E consists of 2-element subsets of V. The number of vertices in G is denoted by $|V|$ and the number of edges is denoted by $|E|$. There are no *self-loops* in undirected graphs, i.e., edges from a vertex to itself. Vertices v and w are *neighbors* or *adjacent* if there is an edge between them. A *path* from v_1 to v_i is a sequence of vertices and edges $v_1, e_1, v_2, e_2, v_3, e_3, ...e_{i-1}, v_i$ such that $e_j = \{v_j, v_{j+1}\}$ for $1 \leq j < i$. The length of a path is the number of edges of that path. Two vertices v_i and v_j are *connected* if there is a path from v_i to v_j. If the two end-points v_1 and v_i are the same, then the path is called a *cycle*. A *forest* is a graph containing no cycles. A *tree* is a connected forest. Two vertices are called *end-points* of an edge if they are connected by that edge. If e is an edge with an

end-point v, we also say that edge e is *incident* with v. A node is called a *leaf* if it is incident with only a single node. The *degree* of a vertex is the number of neighbors of that vertex. A *subgraph* $H = (V', E')$ of $G = (V, E)$ is a graph such that $V' \subseteq V$ and $E' \subseteq E$. A subgraph $H = (V', E')$ of $G = (V, E)$ is *induced* by a vertex subset $V' \subseteq V$ if H contains all edges $\{u, v\} \in E$ for every pair of vertices $u, v \in V'$. A *spanning* subgraph $H = (V', E')$ of a graph $G = (V, E)$ is a subgraph that spans all vertices of G. A *spanning tree* is an acyclic connected graph which contains all vertices of G. Two paths are *edge-disjoint* if they do not have an edge in common, and two paths are *vertex-disjoint* if they do not have an internal vertex in common. A graph G is *k-edge-connected* if it contains at least k edge disjoint paths between every pair of vertices of G, for $k \geq 1$, and a graph is k-vertex-connected if it contains at least k vertex disjoint paths between every pair of vertices of G, for $k \geq 1$. For a connected graph G, a node is said to be an *articulation point* if removal of this node disconnects G. Similarly, an edge is called a *bridge* if its removal from a connected graph G disconnects the graph.

A *directed graph* $G = (V, E)$ is a pair, where V is the set of vertices, and E is the set of edges that consists of ordered pairs of vertices. When vertices or edges in a graph are assigned numerical values, the graph is called a *weighted graph*. Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there is a bijection $f : V \to V'$ such that $(u, v) \in E$ if and only if $(f(u), f(v)) \in E'$. That is, there is a one-to-one correspondence between the vertex sets V and V', and for every edge of G there is an image in G', and vice versa.

Given a graph $G = (V, E)$ with a nonnegative weight $w(u, v)$ assigned to every edge $\{u, v\} \in E$, a *minimum spanning tree* is a tree that spans all vertices of G and has minimum total weight. For a graph G, a subset of its vertices is an *independent set* if no two vertices in the subset are adjacent. A *dominating set* D of a graph G is a subset of the vertices of G in which every vertex not in D is adjacent to at

least one vertex of D. If D induces a connected subgraph, the subset is known as connected dominating set. A *clique* C is a subset of the vertices of G, such that there exists an edge between every pair of vertices in C. A graph is *triangle-free* if no two adjacent vertices are incident to a common vertex. The *chromatic number* of a graph $G = (V, E)$ is the minimum number of colors required to assign color to each vertex of G such that no two adjacent vertices have the same color.

Let M denote a set of points, and $d(x, y)$ denote a distance function for $x, y \in M$. If the following three properties are satisfied, then (M, d) is called a metric space [7].

- $d(x, y) \geq 0$, where equality holds only if $x = y$;
- $d(x, y) = d(y, x)$ for $x, y \in M$;
- $d(x, y) \leq d(x, z) + d(z, y)$ for all $x, y, z \in M$.

An n-dimensional Euclidean space is represented by $(\mathbb{R}^n, d)$, where $\mathbb{R}$ is the set of real numbers and d is the Euclidean distance between any two points. Commonly, the two-dimensional Euclidean space is known as the *Euclidean plane*.

In the following section we describe those classes of graphs which are considered in this thesis to model wireless ad-hoc networks.

2.2 Special Graph Classes

To study wireless ad-hoc networks, simplified graphs are used rather than general graphs. The way nodes of a wireless ad-hoc network are spread in the region together with their neighborhood information usually suggests that the underlying graph could be a geometric object. Therefore, there are several graph models of wireless ad-hoc networks in the literature that are geometric. The most widely used

graph class for wireless ad-hoc networks is the class of *unit-disk graphs*. In a unit disk graph, every node corresponds to a disk of unit diameter in the plane, and two nodes are adjacent if the corresponding disks intersect. Equivalently, one can associate each node with a disk of unit radius of the plane, and define two nodes to be adjacent if the disk corresponding to one node contains the center of the disk corresponding to the other node, i.e., if the Euclidean distance between the centers of the disks is at most 1 [35]. Such a graph is a model of a wireless ad-hoc network consisting of nodes with omnidirectional antennas and equal transmission ranges. The unit radius of each disk represents the transmission range over which the disk can send and receive messages directly. The simple structural properties of unit disk graphs are exploited to achieve good results for various problems arising in wireless ad-hoc networks. In reality, however, it is not necessary that a node always reaches every other node located within its transmission range due to the presence of obstacles like walls, buildings, or signal distortions [4]. For instance, consider the disk in Fig 2.1 representing a wireless node. The shaded area is the area within which the node can send and receive messages, however, the node cannot transmit to the white region. This shows that the transmission range of such a node does not cover the whole area of the disk.

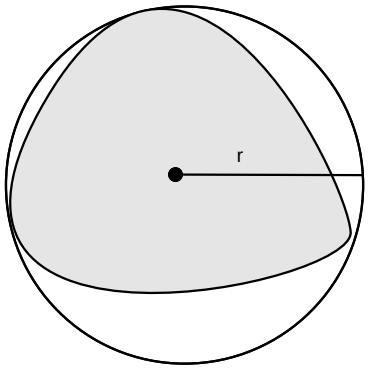


Figure 2.1: Transmission range may not cover the whole disk.

To model the above situation, researchers have considered more general graph

models than unit disk graphs. The class of α-*unit disk graphs*, also called quasi-unit disk graphs, is one of them [4, 35]. In an α-unit disk graph, for $\alpha \leq 1$, nodes are adjacent if the Euclidean distance between them is at most α, non-adjacent if their distance is greater than 1, and can be adjacent or non-adjacent otherwise. This means that a node always reaches another node if both are located at a distance at most α, and two nodes could be non-adjacent even if their Euclidean distance is between α and 1, depending on the circumstances mentioned above. With $\alpha = 1$, an α-unit disk graph is a unit disk graph.

Usually in wireless ad-hoc networks, many nodes are adjacent to each other and the number of independent neighbors of a node is not very large. Due to this property of wireless ad-hoc networks, a class of graphs namely *bounded independence graphs* (BIG) is considered in [49, 34, 33]. Let the r-neighborhood of a node u be the set of nodes that are at most r hops away from u. Bounded independence graphs are the graphs in which the maximum number of independent nodes in the r-neighborhood of any node is bounded by a polynomial $p(r)$ [49]. Inspired by bounded independence graphs, we consider a more general graph class referred to as the class of $(\lambda + 1)$-claw-free graphs. Let us define a t-*claw* in a graph first. A t-*claw* in a graph is an induced subgraph on $t + 1$ nodes that is isomorphic to $K_{1,t}$, i.e., the star with one node in the center and t independent leaves. A $(\lambda + 1)$-claw-free graph is a graph that does not contain a $(\lambda + 1)$-claw. Equivalently, a graph is $(\lambda + 1)$-claw-free if every node has at most λ independent neighbors. Note that for $\lambda = p(1)$, bounded independence graphs are $(\lambda+1)$-claw-free. We mention that the definition of a t-claw-free graph is different from a *claw-free* graph which is known in the literature as a graph that has no induced subgraph isomorphic to $K_{1,3}$, i.e., if no vertex has three pair-wise non-adjacent neighbors (see for example [9]).

In unit disk graphs, the number of independent neighbors of every node is at most 5 [59]. For α-unit disk graphs, we will show later the number of independent

Graph class	value of λ
UDG	5
α-UDG	$4/\alpha^2 + 4/\alpha + 1$
bi-directional-disk graphs	$(18\lceil \log_2 m \rceil + 12)$, for $m > 1$
bounded independence graphs	$p(1)$

Table 2.1: Values of λ for which different graph classes are $(\lambda + 1)$-claw-free.

neighbors of every node is at most $(4/\alpha^2 + 4/\alpha + 1)$. Therefore, unit disk graphs and α-unit-disk graphs are special cases of $(\lambda + 1)$-claw-free graphs.

There is also another graph class employed to model wireless ad-hoc networks known as the class of bi-directional disk graphs. A bi-directional disk graph models a wireless ad-hoc network when nodes have different transmission ranges [53]. Let $G = (V, E)$ be a bi-directional disk graph. Every node $v_i \in V$ corresponds to a point in the Euclidean plane and r_i is the transmission range of a node $v_i \in V$. Two disks v_i, v_j are adjacent if $d(i, j) \leq min\{r_i, r_j\}$ where $d(i, j)$ represents the Euclidean distance from v_i to v_j [53]. In other words, two disk v_i and v_j are adjacent if the Euclidean distance between the two disks is not more than the radius of the disk with minimum transmission range of the two disks. The edge (v_i, v_j) is called a bi-directional edge. See Fig 2.2 for an example. In (a) y is inside the transmission range of x, but x is not in the transmission range of y. Therefore, there will be no edge between these two nodes in a bi-directional disk graph. In (b), x and y are in the transmission range of each other, thus there will be an edge between x and y in a bi-directional disk graph.

Let r_{min} denote the minimum transmission range and r_{max} denote the maximum transmission range. Let $m = r_{max}/r_{min}$. If $m = 1$, this means that the disks are equisized, hence the graph is a unit disk graph. For $m > 1$, the number of independent neighbors of every node is at most $6(3\lceil \log_2 m \rceil + 2)$ [53]. Thus, bi-directional disk graphs are included in the class of $(\lambda + 1)$-claw-free graphs.

See Table 2.1 for an overview of the values of λ for which the graph classes considered are $(\lambda + 1)$-claw-free.

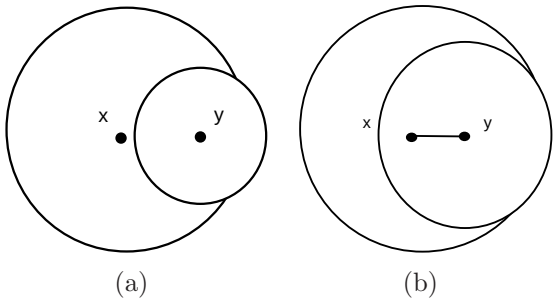


Figure 2.2: (a) y is in the transmission range of x but x is not in the transmission range of y (b) x and y are in each other's transmission range

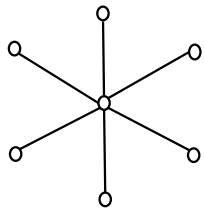


Figure 2.3: Graph G that is α-UDG but not UDG.

By the definitions of the graph classes considered, it follows that UDG $\subseteq$ α-UDG $\subseteq$ BIG $\subseteq$ $(\lambda + 1)$-claw-free. In the following, we show that each of these inclusions is proper.

UDG$\subset$ α-UDG:

Consider the graph in Fig. 2.3 with six independent nodes centered at u, and the distance between every pair of adjacent nodes is 1. The distance between any two consecutive neighbors of u is also 1. We know that the number of independent neighbors of every node in a unit disk graph is at most 5, therefore, G, in which the number of independent neighbors of u is six, cannot be a unit disk graph. However, the graph is an α-unit disk graph if we choose, for example, $\alpha = 0.99$, because we are then free to include or exclude edges between nodes at distance 1. Note that

the number of independent neighbors of u is not more than $4/\alpha^2 + 4/\alpha + 1$ in this graph.

α-UDG $\subset$ BIG:

We show that there is a class of graphs that are bounded independence graphs but not α-unit disk graphs as follows.

It is known that there are triangle-free graphs with arbitrarily large chromatic number. For example, Mycielski's construction [41] can be used to obtain such graphs. Let G_k be a triangle-free graph with chromatic number k, and H_k be the complement of G_k. Let L_k be the graph obtained from H_k by adding a new vertex x and making x adjacent to all vertices of H_k. Consider the class C of graphs that contains L_k for all values of k. C is a class of bounded independence graphs because no graph in C contains an independent set of size 3 (an independent set of size 3 would correspond to a triangle in G_k, a contradiction to G_k being triangle-free).

C is not a class of α-unit disk graphs. Assume for a contradiction that C is a class of α-unit disk graphs for some fixed value of α. Consider any graph L_k and the vertex x in L_k. Every vertex in L_k is a neighbor of x, so all vertices in $L_k - \{x\} = H_k$ are located in a disk of radius 1 with center x. This implies that the vertices of H_k can be partitioned into $O(1/\alpha^2)$ many cliques (because the disk of radius 1 can be covered by $O(1/\alpha^2)$ squares of side length $\alpha/\sqrt{2}$, and the vertices in each such square form a clique). But this means that G_k can be partitioned into $O(1/\alpha^2)$ many independent sets. This cannot hold for all graphs G_k, because the chromatic number of G_k is k and thus exceeds $O(1/\alpha^2)$ for sufficiently large k. So we have a contradiction. This shows that C is not a class of α-unit disk graphs.

BIG $\subset (\lambda + 1)$-claw-free graphs:

Binary trees are included in the class of $(\lambda + 1)$-claw-free graphs but not included in the class of bounded independence graphs. Let us consider a binary tree of depth greater than r (Fig. 2.4). The r-neighborhood of the root node v contains at least 2^r independent nodes, namely the nodes at depth r in the tree. Since we know that in a bounded independence graph, the size of any independent set in the r-neighborhood of every node is bounded by a fixed polynomial in r, binary trees are not included in the class of bounded independence graphs. However, binary trees are $(\lambda + 1)$-claw-free with $\lambda = 3$, as no vertex has more than 3 neighbors.

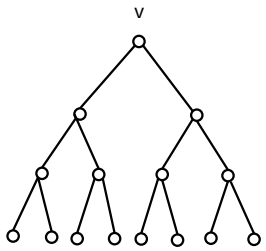


Figure 2.4: A graph that is $(\lambda + 1)$-claw-free but not a bounded independence graph.

2.3 Algorithms and Computational Complexity

An algorithmic problem describes the input and specifies the properties of the desired output [20]. An *instance* of the problem consists of the input specified in the problem statement. The time-complexity of an algorithm is the number of computation steps taken by the algorithm to transform the input into the output, and can be represented as a function of the input size. For example, if we consider graph problems, the input size can either be the size of the vertex set or the edge set in the graph. While designing an algorithm for a problem, the foremost aim is to

solve the problem as efficiently as possible. The efficiency of algorithms is usually judged by their time-complexity. When we analyze the relative performance of two algorithms, we compare their time-complexity functions. The time-complexity is commonly referred to as the worst-case running time of an algorithm also called the *upper bound* on the running time and is represented using the O-notation:

Assume that we have two functions $f(n)$ and $g(n)$. We say that $f(n)$ is $O(g(n))$ if there exists a constant c such that $f(n) \leq c \cdot (g(n))$ for all values of $n \geq n_0$ for some constant n_0. It is also written as $f(n)$=$O(g(n))$.

For example, assume that an algorithm has running time bounded by the value of a function $f(n) = n^3 + n + 2$ for every input. We say that the worst-case running-time is $O(n^3)$. When we consider large enough input size, lower order terms and constants are discarded because their effect is negligible as compared to the higher order terms in the growth of the function (as $n \to \infty$).

For input length n, we say that an algorithm runs in polynomial time if the time complexity of that algorithm is $O(n^k)$ for some constant k. Algorithms are said to be efficient algorithms if their time-complexity is bounded by a polynomial, i.e., $O(n^k)$. When $f(n)$ is $\Omega(g(n))$, it means that there is a constant c such that $f(n) \geq c \cdot g(n)$, for all $n \geq n_0$ for some constant n_0. The notation $\Omega(g(n))$ is often used to specify lower bounds on the running time of a specific algorithm or of all algorithms for a certain problem.

Now let us come to the notion of NP-completeness. We need to consider decision problems first to understand this. For any instance of a decision problem, there are two possible answers, either "yes" or "no". An instance of a decision problem for which the answer is "yes" is referred to as yes-instance, whereas an instance for which the answer is "no" is referred to as no-instance. Let us represent a decision problem by the set of inputs L that contains the yes-instances. L belongs to the complexity class P if there is an algorithm A that has polynomial time

complexity and outputs "yes" for every input $i \in L$ and "no" for every input $i \notin L$. If there is an algorithm A that verifies in polynomial time using a certificate C of polynomial size whether the answer for a given input $i \in L$ is "yes", then the decision problem L belongs to the complexity class NP.

Consider two decision problems Q_1 and Q_2. We say that a decision problem Q_1 *polynomially reduces* to Q_2 if any instance I of Q_1 transforms into an instance I' of Q_2 such that:

- there exists a polynomial-time algorithm which computes I' from I, and
- I is a yes-instance if and only if I' is a yes-instance.

If Q_1 is polynomially reducible to Q_2, then the relative difficulty of two problems is established. We say that Q_2 is at least as hard to solve as Q_1.

A problem Q_1 is in the complexity class NP-hard if any problem Q_2 in NP can be reduced in polynomial time to Q_1.

A problem Q_1 is NP-complete if:

- Q_1 is in NP, and
- for any problem Q_2 in NP, there is a polynomial-time algorithm that reduces any instance of Q_2 into an instance of Q_1.

Some famous examples of NP-complete problems are travelling salesman, vertex-cover, independent set, and Hamiltonian path problem [12]. No efficient algorithm is known for any of the NP-complete problems. If a single NP-complete problem can be solved by a polynomial time algorithm, then all problems in this class can be solved in polynomial time. This idea raises the question of determining whether $P = NP$ or $P \neq NP$. So far, nobody could answer this question in either direction, and in theoretical computer science this is the most famous open problem today.

In contrast to decision problems, optimization problems are the problems that ask to find best feasible solutions. An optimization problem P is defined by a tuple (I, S, val, goal) [2] such that:

- I is the set of instances of P,
- S is a function that maps any instance $i \in I$ to all feasible solutions of i,
- val is a function that assigns a positive integer to each pair (i, s), where $i \in I$ and $s \in S(i)$, and
- goal of the problem P is either maximization or minimization.

An optimization problem can be converted into its decision version and is thus at least as hard to solve as the decision version. In this sense the theory of NP-completeness is also related to the optimization problems.

2.4 Approximation Algorithms

None of the NP-complete problems can be solved optimally in polynomial time unless $P = NP$. If the problem size is small, then an exponential solution can work. However, for sufficiently large input size of a problem, the exponential time-complexity means that this solution is infeasible. Therefore, an alternative way to tackle such problems is to use polynomial-time algorithms that give solutions close to the optimum. These algorithms are known as *approximation algorithms*.

Let $C(A)$ be the cost of the approximate solution produced by an algorithm A, and $C(OPT)$ be the cost of the optimal solution for a given instance of the problem. For a minimization problem P, an approximation algorithm A has performance guarantee or approximation ratio $\alpha \geq 1$ if for any instance of the problem it holds that:

$$C(A) \leq \alpha \cdot C(OPT)$$

For a maximization problem, an algorithm A has an approximation ratio $\alpha \leq 1$ if the following inequality holds for all instances of the problem:

$$C(A) \geq \alpha \cdot C(OPT)$$

Thus, for any instance of a minimization problem, an α-approximation algorithm is a polynomial-time algorithm which outputs a solution of cost at most α times the optimum [57]. Likewise, for a maximization problem, an α-approximation algorithm is a polynomial-time algorithm which gives a solution of cost at least α times the optimum solution for any instance of the problem.

For an optimization problem, a *polynomial-time approximation scheme* ($PTAS$) is an algorithm that takes as input an instance of the problem and a value $\epsilon \geq 0$, and outputs a solution within a factor $1 + \epsilon$ of the optimal in polynomial time in the size of the input [12]. There is a trade-off between the running time of the algorithm and the value of epsilon; as the value of ϵ gets smaller, the running time of the algorithm grows faster. An optimization problem belongs to the class APX, if there exists a constant-factor approximation algorithm for the problem. There is a notion of approximation-preserving reductions (see, e.g., [2] for a definition of AP-reductions). A problem Q_1 is APX-hard if, for any problem $Q_2 \in APX$, Q_2 can be reduced to it with an approximation-preserving reduction. An APX-hard problem is APX-complete if it belongs to to the class APX. If a problem is APX-complete, then it does not admit $PTAS$ unless $P = NP$ [2].

There are several ways in the literature to design approximation algorithms for NP-complete problems, for example, greedy algorithms, linear programming, the primal-dual method and dynamic programming [2, 55]. The existing approximation algorithms that we use as black-box in our work are based on linear programming, therefore, here we introduce the technique of linear programming. A linear program

formulates an optimization problem that asks to minimize or maximize a linear objective function subject to linear constraints. Given two n-dimensional vectors c and x, an m-dimensional vector b, and an $m \times n$ matrix A, an objective function to be minimized (or maximized) is given by $c^T x$, where c^T denotes the transpose of c [12]. $c^T x$ is the inner product of two vectors whereas Ax is a matrix-vector product. The constraints that are required to be satisfied are $Ax \geq b$ and $x \geq 0$. When all constraints are satisfied by a solution, the solution is called a *feasible* solution. When the value of the objective function $c^T x$ is minimum (or maximum) over all feasible solutions, the solution is called an *optimal* solution.

In contrast to linear programming, in integer programming each variable can take an integer value only. Solving an integer program is an NP-hard optimization problem, but a linear program can be solved optimally in polynomial time. For the purpose of designing an approximation algorithm for an optimization problem, the idea is to formulate an integer program first, then relax the integer constraints to get an LP and solve the LP to get an optimal fractional solution, and finally, construct an integral solution from the LP's optimal solution using some rounding techniques.

As an example, we show how the generalized Steiner network problem is modelled as an integer program in Jain's algorithm [26], which will be discussed briefly in Chapter 3. First we define the generalized Steiner network problem: given an undirected graph $G = (V, E)$ with nonnegative cost c_e on every edge $e \in E$, and a connectivity requirement $r_{i,j}$ for all pairs of vertices i, j, compute a subgraph $H(V, E')$ of G of minimum cost such that there are at least $r_{i,j}$ edge-disjoint paths between every pair of vertices i, j. Consider an instance of this problem given by an undirected graph $G = (V, E)$ with nonnegative cost c_e for all $e \in E$, and a connectivity requirement $r_{i,j}$ for very pair of vertices i, j. For all cuts $S \subseteq V$, let us denote the set of edges in E that have only one endpoint in S by $\Delta(S)$. A subset

E' of E is feasible if and only if $|\Delta(S) \cap E'| \geq r_{i,j}$ for all cuts S and $i \in S, j \notin S$. The integer program is then as follows, using a variable $x_e \in \{0,1\}$ for every $e \in E$ that represents whether e is included in the solution or not:

$$\begin{aligned} \text{Minimize} \quad & \sum_{e \in E} x_e c_e & & \\ \text{subject to} \quad & \sum_{e \in \Delta(S)} x_e \geq \max_{i \in S, j \notin S} r_{i,j}, & & \forall S \subseteq V \\ & x_e \in \{0,1\}, & & \forall e \in E \end{aligned}$$

The corresponding linear programming relaxation is given by:

$$\begin{aligned} \text{Minimize} \quad & \sum_{e \in E} x_e c_e & & \\ \text{subject to} \quad & \sum_{e \in \Delta(S)} x_e \geq \max_{i \in S, j \notin S} r_{i,j}, & & \forall S \subseteq V \\ & 0 \leq x_e \leq 1, & & \forall e \in E \end{aligned}$$

Jain's algorithm [26] then solves this linear programming relaxation without constructing the whole linear program explicitly (as the number of constraints is not polynomial), and applies iterative rounding.

For details on the preliminaries discussed in this chapter, we refer to [13, 22] for concepts on graphs, [20] for the theory of NP-completeness, and [57, 2, 55] for approximation algorithms.

Chapter 3

Literature Review

We begin this chapter with some basics of Steiner trees. In the rest of the chapter, we discuss work related to the optimization problems we study in this thesis.

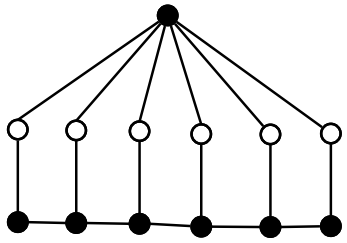
3.1 Basics of Steiner trees

Motivated by communication in wireless ad-hoc networks, we consider multicast routing in which one node wants to send messages to multiple receivers via multi-hop transmissions. A wireless network can be modelled as a graph, and the multicast communication can be achieved by a multicast tree in the form of a Steiner tree. We can refer to the sender and the receivers as *terminals*. Apart from terminals, there are other nodes in a Steiner tree which only take part in the transmission to forward messages but are not the intended receivers. Such nodes are referred to as *Steiner nodes*. So far, many variants of the Steiner tree problem have been studied in the literature. For the Steiner tree problem in graphs, one distinguishes the edge-weighted version where the goal is to minimize the total weight of the edges of the Steiner tree, and the node-weighted version where the goal is to minimize the total weight of the Steiner nodes. Both versions have also been studied

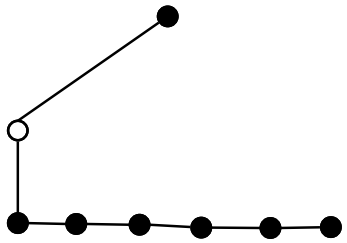
in the un-weighted version, where the goal is to minimize the number of edges or the number of Steiner nodes. Note that a Steiner tree has a minimum number of edges if and only if it has a minimum number of Steiner nodes. For the study of approximation algorithms, however, the two objective functions are very different, and this can be explained with the help of the following example.

Consider the graph G in Fig 3.1, and Steiner trees for this graph. Let G' be the optimal solution, and G'' an approximate solution for the given graph G. Black nodes represent terminals and white nodes are Steiner nodes. There are two rows in G in which the number of nodes is equal, and there is a single node that is located at the top of these two rows. In the bottom row of G, assume that the number of terminals is k. Thus, the total number of nodes in G is $2k + 1$. Now, let us look at the approximation ratio of G'' with respect to the number of edges, and also with respect to the number of Steiner nodes. The number of edges is $2k$ in G'', and $k + 1$ in G'. So, G'' has approximation ratio $\frac{2k}{k+1}$ with respect to the number of edges which is arbitrarily close to 2 for $k \to \infty$. The total number of Steiner nodes is k in G'', and 1 in G'. Therefore, G'' is a k-approximate solution with respect to the number of Steiner nodes. We can see that a good approximate solution for the objective of minimizing the number of edges of a Steiner tree is not guaranteed to be an equally good approximate solution for the objective of minimizing the number of Steiner nodes. Hence, minimizing the number of Steiner nodes, and minimizing the number of edges, are different objective functions from the perspective of approximation algorithms.

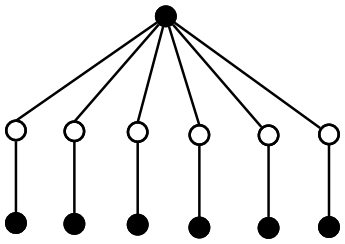
The single-pair shortest path problem and minimum spanning tree problem are special cases of the Steiner tree problem [46]. Given a graph $G = (V, E)$, weight w_e on every edge $e \in E$, and a pair of vertices s, t, the single-pair shortest path problem asks to find a path of minimum total weight between the two vertices s and t. Given a graph $G = (V, E)$, $K \subseteq V$, and weight w_e assigned to every edge



(a) G



(b) G' optimal solution w.r.t number of edges and number of Steiner nodes.



(c) G''

Figure 3.1: An example to illustrate that minimizing the number of Steiner nodes is different from minimizing the number of edges of a graph in terms of approximation algorithms. Black nodes represent terminals, white nodes are Steiner nodes. Approximation ratio of G'' w.r.t. number of edges is 2, and k w.r.t. number of Steiner nodes.

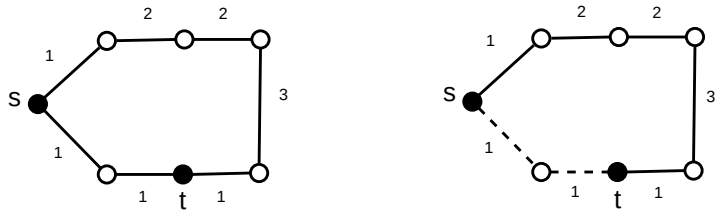


Figure 3.2: Shortest path shown with dashed lines

$e \in E$, if K contains only a single pair of vertices, then the Steiner tree problem reduces to the shortest path problem for the given pair of vertices (Fig 3.2). Given a graph $G = (V, E)$, weight w_e on every edge $e \in E$, the *minimum spanning tree problem* asks to find a tree that spans every vertex of G and has minimum total weight. Given a graph $G = (V, E)$, $K \subseteq V$, and weight w_e assigned to every edge $e \in E$, if $K = V$, then the Steiner tree problem reduces to the minimum spanning tree problem (see Fig 3.3).

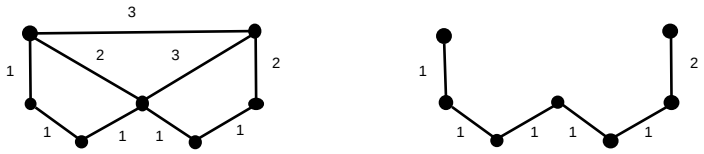


Figure 3.3: $K = V$, Minimum spanning tree is equivalent to Steiner tree in this example.

3.2 Related Work on Steiner Trees

The node-weighted Steiner tree problem is studied in [30, 25], and the edge-weighted Steiner tree problem in [36, 28, 51, 61, 47, 8]. For a long time the best known

approximation algorithm for the edge-weighted Steiner tree problem in general graphs was that of Robins and Zelikovsky [47], which achieved approximation ratio 1.55. Recently, Byrka et al. [8] achieved a better approximation ratio for this problem which is 1.39. Their algorithm is based on linear programming. For the node-weighted Steiner tree problem in general graphs, an approximation ratio of $1.35 \ln k$ has been shown by Guha and Khuller [25], where k is the number of terminals.

Besides the Steiner tree problem in graphs, there have also been studies of geometric Steiner tree problems. In the Euclidean Steiner tree problem, the terminals are points in the Euclidean plane, the length of an edge is the Euclidean distance between its endpoints, and the goal is to minimize the total edge length of the tree [14]. Note that arbitrary points can be used as Steiner nodes. The rectilinear Steiner tree problem is the Euclidean Steiner tree problem in the plane with the additional constraint that all edges of the Steiner tree must be horizontal or vertical. Garey and Johnson showed that the rectilinear Steiner tree problem is NP-hard [19]. It follows from a similar reduction presented in the proof of Theorem 7.1 that for unit disk graphs, the problem of computing a Steiner tree with a minimum number of edges (or, equivalently, with a minimum number of Steiner nodes) is NP-hard [11].

Xu et al. [60] have considered the Steiner tree problem in unit disk graphs with the goal of minimizing the number of internal nodes of the Steiner tree. The motivation for this objective function is that the internal nodes of the Steiner tree are the nodes that need to actively forward messages, while the leaves are only receivers. They call this the *Minimum Steiner Connected Dominating Set* problem and present an algorithm with approximation ratio $2\rho + 1$, where ρ is the best known approximation ratio of the edge-weighted Steiner tree problem in graphs, currently $\rho = 1.39$. A main difference between their objective function and ours

is that terminals that are internal nodes of the Steiner tree count towards their objective but not towards ours (because we only consider the cost of the Steiner nodes). Therefore, approximation results for the two problems do not translate to each other. Furthermore, Xu et al. [60] only consider the un-weighted version of their problem.

Zou et al. [62] consider the node-weighted Steiner tree problem in unit disk graphs and present a 2.5ρ-approximation algorithm, where ρ is the best known approximation ratio for the edge-weighted Steiner tree problem in arbitrary graphs. With $\rho = 1.39$ [8], this gives approximation ratio 3.475 for node-weighted Steiner trees in unit disk graphs. To obtain this result, Zou et al. [62] define the weight of an edge to be the sum of the weights of the two endpoints and then apply an algorithm for the edge-weighted Steiner tree problem. In this thesis, we follow the same approach for the node-weighted Steiner tree problem, but our analysis is more general and extends to classes of graphs such as quasi-unit disk graphs, bounded independence graphs, and $(\lambda + 1)$-claw-free graphs.

There is also a study of the node-weighted minimum Steiner tree problem in line of sight networks presented by Frieze et al. [18]. The authors refer to this problem as "Relay Placement problem". In a line of sight network, every node has a transmission range r, and nodes are located at intersection points of rows and columns. Two nodes are visible to each other if they are in the same row or same column, and at a distance at most r from each other. Frieze et al. [18] present a ρd-approximation algorithm for the node-weighted minimum Steiner tree problem, where ρ is the approximation ratio of the edge-weighted version of the Steiner tree problem and d is the maximum degree of a spanning tree in the line of sight graph model. A graph having a spanning tree of maximum degree d for every connected component is referred to as d-cohesive in [18]. Our approach is similar to theirs but is significantly more general since we extend our study to several classes of graphs

employed to model wireless ad-hoc networks.

For routing in wireless networks, one approach is to use a virtual backbone comprising a set of nodes of the network, and connect the remaining nodes of the network to this backbone. The work of Min et al. [39] is based on this idea. In wireless networks, a virtual backbone could be constructed with the help of a connected dominating set. It is desirable that the size of the virtual backbone is small. Therefore, the goal is to find a connected dominating set of minimum size. It has been shown that the problem of computing a minimum connected dominating set is NP-hard even for unit disk graphs, hence several approximation algorithms are proposed for this problem in the literature. Min et al. [39] first compute a dominating set, and then given the set of nodes of the dominating set as terminals, the idea is to construct a Steiner tree connecting terminals with the help of a minimum number of nodes that are not terminals. Their algorithm yields approximation ratio 6.8 for connected dominating sets in unit disk graphs.

Node-weighted Steiner trees have also been used in the algorithm presented by Liang [37] for finding minimum energy multicast trees in the bi-directional disk graph model for wireless ad hoc networks. The minimum energy multicast tree problem is defined as follows: Given a wireless network $Q = (N, L)$, a source node s and a subset of destination nodes $D \subseteq (N - \{s\})$, the minimum-energy multicast problem is to compute a multicast tree rooted at s that spans the vertices in D such that the sum of transmission powers of all nodes except leaves is minimized. It is assumed that leaves do not participate in transmitting packets, therefore, their weight is not included in the objective function. An undirected node-weighted graph G is constructed for the network Q. The set of terminals for G corresponds to s and D. For this instance of the node-weighted minimum Steiner tree problem, a tree T is computed with an approximation algorithm for the node-weighted minimum Steiner tree problem with best known approximation ratio α. Then the

tree is modified repeatedly to get a multicast tree in which the total transmission power of the nodes is minimized approximately. The algorithm presented by [37] achieves an approximation ratio which is within a constant factor of α.

Panigrahi [45] has recently addressed the minimum Steiner activation network problem for wireless networks. Given a graph $G = (V, E)$, a collection of subsets of terminals $T_1, T_2, ...T_k \subseteq V$, a parameter x_v chosen from a set X of possible values at every node $v \in V$, and the activation function $f_{u,v}$ of every edge $u, v \in E$. The activation function is a mapping $f_{u,v} : X \times X \rightarrow \{0, 1\}$. Depending on the chosen values x_u and x_v, if the edge u, v is active for transmissions, the activation function gives 1, and otherwise 0. In the *minimum Steiner activation network* problem (MSAN), every pair of terminals in each of the k sets $T_1, T_2, ...T_k \subseteq V$ is aimed to be connected in the activated subgraph and the goal is to minimize the sum of the values of x_v. The MSAN problem is a generalization of the node-weighted minimum Steiner network problem [45] defined as follows: Given a graph $G = (V, E)$, nonnegative node-weights, and subsets of terminals $T_1, T_2, ...T_k$, the objective is to compute a minimum-cost forest that contains all subsets of terminals, and every pair of terminals in a subset T_i is connected. Panigrahi [45] reduces the given instance of the MSAN into an instance of the minimum node-weighted Steiner network problem and then uses the algorithm by Klein and Ravi [30]. This yields approximation ratio $O(\log n)$ for the MSAN problem. The node-weighted minimum Steiner tree problem is a special case of the minimum Steiner network problem, so the MSAN problem also generalizes the node-weighted minimum Steiner tree problem.

In several studies, heuristic solutions have been proposed for constructing multicast trees where the performance is tested with the help of simulations. For example, Wieselthier et al. [56] worked on establishing multicast trees when the transmission power of nodes is adjustable. When a node uses high transmission

power, it can reach a large number of destination nodes. However, when the transmission power of a node is lower, it does not reach as many nodes and, therefore, needs to transmit over longer paths to the destination nodes. Let v and w be inside the transmission range of source u. While transmitting packets to v, w will also hear the transmission of u.

This property of wireless networks which enables multiple nodes to receive messages simultaneously is known as *wireless multicast advantage*. In the method presented by [56], first a tree is constructed using the broadcast incremental power method: Start with a source node, add one node at a time to the tree (such that the increase in transmission power of the parent of the new node is minimized) until all nodes are included in the tree. In the second step, nodes that are not used to connect the intended multicast receivers are deleted from the tree. In this process, those nodes whose distant neighbors are deleted reduce their transmitting power level too. This multicast incremental power technique which benefits from the wireless multicast advantage shows better results than other techniques used to construct multicast trees that do not consider the wireless multicast advantage [56].

We refer the interested reader to [58, 48, 52, 3] for simulation-based studies on multicast communication in wireless ad-hoc networks.

3.3 Fault-tolerant Steiner Problems

The problems $NW2ECS$ and $NWkECS$ studied in this thesis are related to the classical combinatorial problem known as *generalized Steiner network problem* or *survivable network design problem* (SNDP), defined as follows: Let an undirected graph $G = (V, E)$ with vertex set V and edge set E be given. For every edge $e \in E$, there is a nonnegative weight c_e, and for every pair u, v of vertices in V,

there is a connectivity requirement $r_{u,v}$. The objective is to find a minimum-cost subgraph such that there are at least $r_{u,v}$ edge-disjoint paths between u and v, for all $u, v \in V$ [24]. There are several special cases of this problem including the 2-edge-connected Steiner subgraph problem, and the k-edge-connected Steiner subgraph problem. In the edge-weighted version of the 2-edge-connected Steiner subgraph problem, $r_{u,v}$ equals 2 if $u, v \in K$ and 0 otherwise. Similarly, for the special case of the k-edge-connected Steiner subgraph problem, $r_{u,v}$ equals k if $u, v \in K$ and 0 otherwise. A 2-approximation algorithm for the generalized Steiner network problem was presented by Jain [26]. Hence, there is a 2-approximation algorithm for the edge-weighted version of the 2-edge-connected Steiner subgraph problem and k-edge-connected Steiner subgraph problem for arbitrary graphs.

For a graph $G = (V, E)$, weights w_v for all $v \in V$, and edge-connectivity requirement $r_{u,v}$ for all $u, v \in K \subseteq V$, Nutov [43] studied the problem of computing a minimum weight subgraph H of G that contains K and has at least $r_{u,v}$ edge-disjoint paths between every pair of terminals. The algorithm presented in [43] yields approximation ratio $r_{max} \cdot O(\ln |K|)$, where $r_{max} = \max_{u,v \in K} r_{u,v}$ and $|K|$ denotes the number of terminals.

There are also problem variants concerned with node-disjoint instead of edge-disjoint paths, for example, edge-weighted and node-weighted k-vertex-connected Steiner subgraphs. The best known approximation ratio for the k-vertex-connected Steiner subgraph problem with respect to edge weights is $O(k^3 \log n)$ [10], and for node-weights, it is $O(k^4 \log^2 n)$ [44], where n is the number of vertices. In this thesis, we consider the 2-vertex-connected Steiner subgraph problem with respect to node weights. For the edge-weighted 2-vertex-connected Steiner subgraph problem, the best known approximation ratio is 2 [17]. For the node-weighted 2-vertex-connected Steiner subgraph problem, the best known approximation ratio is $O(\ln n)$ [43]. A special case of the k-vertex-connected Steiner subgraph problem is the problem

of computing k-vertex-connected spanning subgraphs, i.e., $r_{u,v}$ equals k for all $u, v \in V$. The k-vertex-connected spanning subgraph problem [32, 16] asks to compute a subgraph that contains all vertices of G, and in which every pair of vertices has at least k vertex-disjoint paths. The goal of the k-vertex-connected spanning subgraph problem is to compute a subgraph of minimum total weight.

There is also another version of the generalized Steiner network problem considered in the literature: instead of requirements $r_{u,v}$ for every pair of vertices, an integer value r_u is assigned to nodes [23]. The goal is to find a minimum cost graph satisfying the requirement $r_{u,v} = \min(r_u, r_v)$.

Kamma and Nutov [27] studied survivable networks with the objective of using a minimum number of Steiner points in unit disk graphs. Given a finite set of points $V \subset Z$ in a metric space represented by (Z, d), we have an induced unit disk graph G on V, and there is an edge between every pair of vertices $u, v \in V$ if the distance between them is at most 1. An integer connectivity requirement $r = \{r_{u,v} : u, v \in X \subseteq V\}$ is satisfied if there are r vertex-disjoint paths between every pair of vertices $u, v \in X$. The goal is to find a set of Steiner nodes $S \subset Z \setminus V$ of minimum size such that the new graph G' on $(V \cup S)$ satisfies r (SN-MSP). They reduce an instance of their problem into an instance of the edge-weighted survivable network design problem (SNDP). They prove that an α-approximation algorithm for the instance of SNDP gives an $\alpha \cdot O(k^2)$-approximation algorithm for the SN-MSP instance, where $k = \max_{u,v \in V} r_{u,v}$. There is similarity in their approach and ours as for the 2-vertex-connected Steiner subgraph problem, we transform the given instance into an edge-weighted version of the 2-vertex-connected Steiner subgraph problem, and then solve that instance with an α-approximation algorithm for the edge-weighted version of the problem. We aim to find 2 vertex-disjoint paths whereas their connectivity requirement is more general seeking k vertex-disjoint paths. They consider unit disks in a metric space, however, we consider

unit disk graphs in the Euclidean plane. Our result extends to more general classes of graphs including α-unit-disk graphs, bi-directional disk graphs, and $(\lambda+1)$-claw-free graphs.

Bredin et al. [6] studied fault-tolerance in wireless networks. If the given network is not k-vertex-connected, their algorithm deploys additional nodes to make the network k-vertex-connected. Their algorithm also works in scenarios when several nodes fail and the network connectivity is lost. So, additional nodes are installed to repair the network. Their goal is to minimize the number of additional nodes deployed in the network for k-vertex-connectivity. The network is modelled by a unit disk graph. Given a graph $G = (V, E)$ and connectivity requirement k, the algorithm computes a complete graph $G' = (V, E')$ on the vertex set V, and assigns weight $w(u, v) = \lceil d(u, v) \rceil - 1$ to each edge $(u, v) \in E$. Then, their algorithm constructs a k-vertex-connected subgraph H' for G' of minimum weight using an α-approximation algorithm for the k-vertex-connected Steiner subgraph problem on complete graphs. In the next step, for every edge selected in the k-vertex-connected subgraph of G', new nodes are inserted in G'. Let the new graph obtained be H. There can be two cases of fault-tolerance [6]: (i) when only the original vertices in V are required to be k-vertex-connected, and, (ii) when all vertices in H are required to be k-vertex-connected. Their algorithm ensures that one of the above two cases are satisfied. For the first case, $w(u, v)$ nodes are inserted in H along each edge in H' such that they are at a unit distance from each other. For the second case, $w(u, v)$ groups of k nodes are placed along each edge $\{u, v\}$ in H', and the distance between each group is one unit. Their algorithm is an $O(k^4\alpha)$ approximation algorithm. The step in which they transform their input graph into an edge-weighted complete graph, and then run an existing α-approximation algorithm for the edge-weighted k-vertex-connected subgraph problem, is somehow analogous to our algorithm for the 2-vertex-connected Steiner subgraph problem.

We transform an instance of the node-weighted 2-vertex-connected Steiner subgraph problem to an instance of the edge-weighted version of the problem, and then run an α-approximation algorithm for the 2-vertex-connected Steiner subgraph problem with respect to edge-weights. We minimize the weight of the resulting graph requiring that the property of 2-vertex-connectivity is satisfied, while their objective is to include a minimum number of additional nodes (Steiner nodes) to connect terminals through k vertex-disjoint paths.

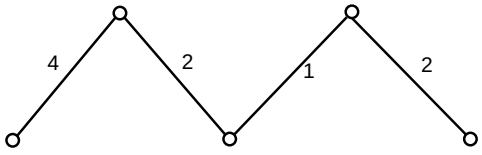
There are also studies of group Steiner problems in the literature. Given an undirected graph $G = (V, E)$, costs c_e for all $e \in E$, and a collection of groups of vertices $K_1, ..., K_M \subseteq V$, the group Steiner tree problem asks to compute a tree that is a subgraph H of G of minimum-cost and contains at least one vertex from each group [21]. Fault-tolerant variants of this problem are studied in [29] including the node-weighted version of the 2-edge-connected group Steiner subgraph problem (called *EC-FTGS*-2 in [29]). The best known approximation ratio for the node-weighted 2-edge-connected group Steiner subgraph problem is $O(\log n)$ [29]. For the edge-weighted version of the problem, the best known approximation ratio is $(2 + \alpha)$, where α is the best known ratio for the Steiner tree problem [29].

3.4 Inapproximability Results of Steiner Problems in General Graphs

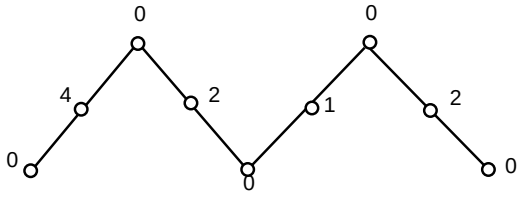
The edge-weighted Steiner tree problem is APX-hard [5]. For the node-weighted Steiner tree problem, no approximation algorithm better than $O(\ln n)$ can be achieved unless $NP \subseteq DTIME(n^{polylog(n)})$[2] because the set-cover problem can be reduced (in an approximation-preserving way) to the node-weighted Steiner tree

[2] $DTIME(n^{polylog(n)})$ represents the class of problems for which there exists a deterministic time algorithm that runs in time $n^{polylog(n)}$.

problem [30]. Since the node-weighted k-edge-connected Steiner subgraph problem generalizes the node-weighted Steiner tree problem, the k-edge-connected Steiner tree problem is also set-cover hard. The node-weighted group Steiner subgraph problem generalizes the node-weighted Steiner tree problem [29] , therefore, we cannot achieve approximation ratio better than $O(\ln n)$ for our 2-edge-connected group Steiner subgraph problem either. For the edge-weighted k-vertex-connected Steiner subgraph problem, one cannot achieve approximation ratio better than $2^{\log^{1-\epsilon} n}$ for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{polylog(n)})$ [31]. The same inapproximability result applies to the node-weighted k-vertex-connected Steiner subgraph problem as the node-weighted version is more general than the edge-weighted version (see Fig. 3.4. Any instance of the edge-weighted problem can be reduced to an instance of the node-weighted version by dividing every edge of the given instance and then assigning weight associated with each divided edge to the corresponding new node [30]). See Table 3.1 for an overview of the known inapproximability results.



(a) An instance of the edge-weighted problem I.



(b) the node-weighted instance I' reduced from I.

Figure 3.4: An instance of the edge-weighted version reduced into the node weighted version.

Steiner tree problem	$\Omega(\ln n)$
k-edge-connected Steiner subgraph problem	$\Omega(\ln n)$
k-vertex-connected Steiner subgraph problem	no $2^{\log^{1-\epsilon} n}$-approximation for any $\epsilon > 0$
2-edge-connected group Steiner subgraph problem	$\Omega(\ln n)$

Table 3.1: Inapproximability of node-weighted Steiner problems in general graphs.

Chapter 4

General Problem

In this chapter we study the node-weighted δ-Steiner subgraph problem. Given a graph $G = (V, E)$, and a subset $K \subseteq V$, the node-weighted δ-Steiner subgraph problem asks to compute a Steiner subgraph $H \subseteq G$ of minimum vertex cost, such that a property $\delta(H, K)$ holds. A Steiner subgraph H contains the vertices of K (terminals), and some vertices in $V \setminus K$ (Steiner nodes). Interestingly, the result we obtain can be applied to the family of problems outlined in Chapter 1, which implies that the node-weighted δ-Steiner problem is a generalization of those problems. We introduce a property P_d^δ in Section 4.3, on the basis of which we achieve a $0.5d\rho$-approximation algorithm for the node-weighted δ-Steiner problem for classes of graphs that satisfy P_d^δ, where ρ is the approximation ratio of the best known approximation algorithm for the edge weighted version of the problem, and d is the maximum degree bound of the solution subgraph.

4.1 Node-weighted δ-Steiner Subgraph Problem

We consider properties $\delta(H, K)$ where H is a graph and K is the set of terminals. Intuitively, the property $\delta(H, K)$ represents that H contains all vertices from K

and satisfies certain connectivity requirements. A graph that satisfies $\delta(H, K)$ is referred to as *δ-Steiner subgraph*. Examples of subgraphs for suitable choice of δ are Steiner subgraphs in which all terminals are connected, 2-edge-connected Steiner subgraphs, 2-vertex-connected Steiner subgraphs, and k-edge-connected Steiner subgraphs.

For a fixed property δ, we define the *node-weighted δ-Steiner subgraph problem* as follows:

Problem 1. *Given a graph $G = (V, E)$, a subset of vertices $K \subseteq V$, and nonnegative weights w_u for $u \in V$, the node-weighted δ-Steiner subgraph problem is to find a subgraph H of G such that the property $\delta(H, K)$ is satisfied. The objective of the node-weighted δ-Steiner subgraph problem is to minimize the sum of the weights of the vertex set of H.*

Besides the node-weighted δ-Steiner subgraph problem, there is an edge-weighted version of the problem defined as follows: Given a graph $G = (V, E)$ with edge costs c_e for $e \in E$, and a subset $K \subseteq V$, compute a Steiner subgraph $H \subseteq G$, such that the property $\delta(H, K)$ holds. The goal of the edge-weighted δ-Steiner subgraph problem is to minimize the sum of the costs of the edges in H.

As our motivation is wireless ad-hoc networks, we present approximation algorithms for the node-weighted δ-Steiner subgraph problem in those graph classes which are employed to model wireless ad-hoc networks. For general graphs, the approximation algorithms proposed for the edge-weighted δ-Steiner subgraph problem in the literature have better approximation factor as compared to the results achieved for the node-weighted version. We show that using the best known ρ-approximation algorithms for the edge-weighted version of the problem in general graphs, we can get better approximation results for the node-weighted δ-Steiner subgraph problem in restricted graph classes.

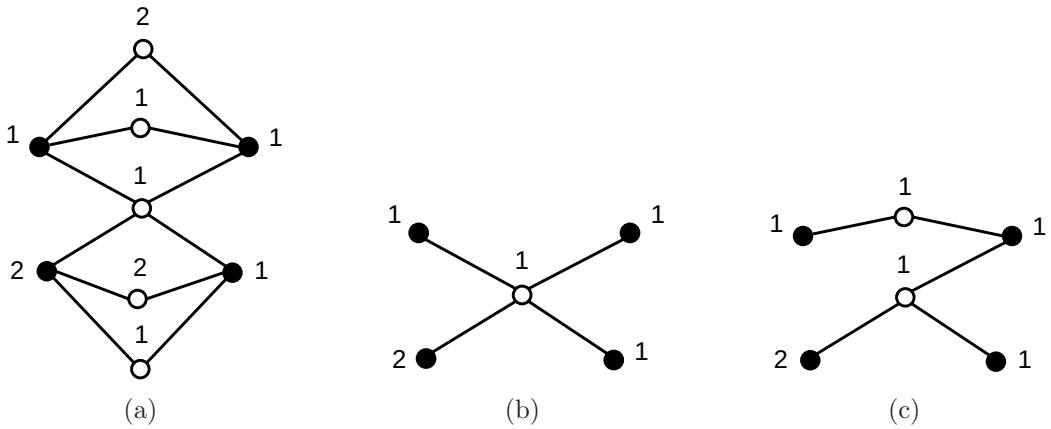


Figure 4.1: Black nodes are terminals and white node are Steiner nodes (a) Graph G with arbitrary weights assigned to terminals (b) Optimal solution for connecting terminals in G (c) solution produced by an algorithm for connecting terminals in G with approximation ratio 7/6.

We assume that terminals have weight zero because they are present in every solution. If we allow the weight of terminals to be more than zero, we can only get a better approximation ratio. For example, see Fig. 4.1, terminals are assigned arbitrary nonnegative weights, whereas in Fig. 4.2 all terminals are assigned weight zero. Clearly, the ratio between approximate solution and optimal solution is 7/6 in Fig. 4.1 which is less than the approximation ratio shown in Fig. 4.2, i.e., 2. Thus, assigning weight more than zero to terminals does not worsen the approximation ratio but may improve it.

Before we present a formal proof for what we have shown in the above example, we state the following inequality that holds if $a \geq 0, b \geq 0, a \geq b$:

$$\frac{a+k}{b+k} \leq \frac{a}{b}, \qquad \forall k \geq 0 \tag{4.1}$$

Lemma 1. *A ρ-approximation algorithm A for instances of the node-weighted δ-Steiner subgraph problem with terminals of weight* 0 *gives also a ρ-approximation algorithm for instances of the node-weighted δ-Steiner subgraph problem with ter-*

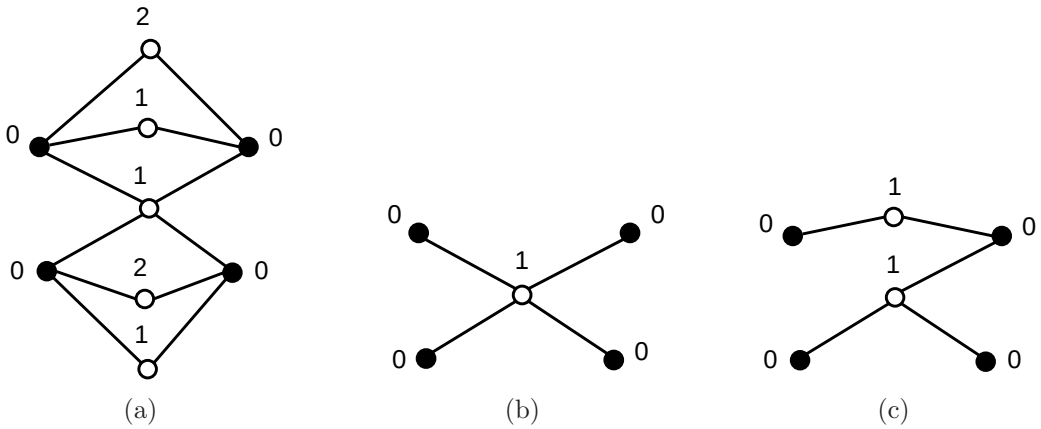


Figure 4.2: Black nodes are terminals and white node are Steiner nodes (a) Graph G with zero weight assigned to each terminal (b) Optimal solution for connecting terminals in G (c) solution produced by an algorithm for connecting terminals in G with approximation ratio 2.

minals of arbitrary non-negative weights.

Proof. Consider an instance of the node-weighted δ Steiner subgraph problem I with arbitrary terminal weights. Let OPT be the weight of optimal solution for I, and $w(K)$ be the weight of terminals in I. We modify I by setting the terminal weights to 0 and denote the modified instance by I_0. Consider the weight of an optimal solution for I_0 and denote it by OPT_0. Let the weight of the Steiner nodes of the solution produced when A is applied to I_0 be $w(S)$.

Observe that: $OPT = OPT_0 + w(K)$, and

$$w(S) \leq \rho \cdot w(OPT_0)$$

The approximation ratio of the solution produced by A for I_0, if taken as solution to I, is then: $\frac{w(S)+w(K)}{OPT} = \frac{w(S)+w(K)}{OPT_0+w(K)}$

Due to inequality (4.1), we get

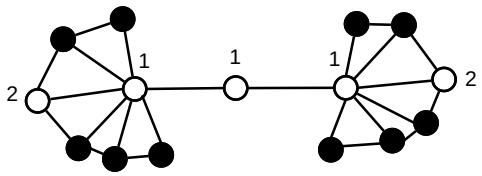
$$\frac{w(S)+w(K)}{OPT_0+w(K)} \leq \frac{w(S)}{OPT_0} \leq \rho,$$

where $w(K) \geq 0$, $w(S) \geq OPT_0$, and $w(S) \geq 0$, □

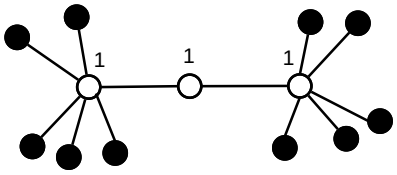
Definition 1. *We say that the property $\delta(H, K)$ is monotone if neither adding an edge to H nor deleting a Steiner node of degree 1 from H, violates the property.*

A Steiner node of degree 1 in a solution means that it is a leaf node and not an intermediate node which is helping to connect other nodes in that solution. For instance, consider a property $\delta(H, K)$ that expresses that H is a Steiner subgraph in which every pair of terminals is connected. Let us assume that there exist Steiner nodes of degree one in H. Even after removing such Steiner nodes, H is a Steiner subgraph that connects all terminals. If we consider a $\delta(H, K)$ such that H is a Steiner subgraph satisfying some higher connectivity requirement, in that case too, a Steiner node of degree 1 can be deleted without harming the connectivity of other nodes in the solution. Thus, the property $\delta(H, K)$ still holds if we remove Steiner nodes of degree 1.

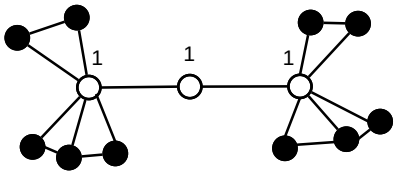
We prove our result for those classes of graphs for which there exist δ-Steiner subgraphs of bounded maximum degree. Due to this reason, it is necessary that the solution subgraph belongs to the same class of graphs to which the input graph belongs. However, the solution we get may not always belong to the class of the input graph. For example, consider the unit disk graph $G = (V, E)$ in Fig 4.3. Black nodes are terminals and white nodes represent Steiner nodes. We wish to compute a Steiner subgraph for G that connects all terminals. One possible solution is the subgraph H^*, which is indeed an optimal solution for this example. However, H^* does not belong to the class of unit disk graphs because no vertex in a unit disk graph can have more than 5 independent neighbors. For the solution to belong to the class of unit disk graphs, we compute the graph H' induced by vertex set $V(H^*)$. The desired connectivity of nodes does not decrease in H' when edges are added to H^* to obtain H'. Therefore, H' satisfies $\delta(H', K)$, and we can achieve our approximation results for the considered graph classes that admit solution subgraphs of bounded maximum degree.



(a)



(b)



(c)

Figure 4.3: (a) Given graph G (b) Solution H^* to G (c) Adding edges to H^* to obtain an induced subgraph H' does not violate δ.

4.1.1 Properties δ for different types of Steiner subgraphs

There are other properties considered in this thesis which are special cases of the property δ. The node-weighted Steiner subgraphs satisfying those special cases of δ are as follows:

Node-weighted connected Steiner subgraphs

We consider the property $\mu(H, K)$ which expresses that H is a graph that contains all vertices in K such that there is a path between every pair of vertices in K. Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for all $v \in V$, and a subset of nodes $K \subseteq V$, a subgraph H for G and K that satisfies the property $\mu(H, K)$ is referred to as node-weighted connected Steiner subgraph. When a Steiner subgraph connects vertices of K and is acyclic, the subgraph is called a Steiner tree, i.e., there is only one path between every pair of terminals. A connected Steiner subgraph can be transformed into a Steiner tree by removing those edges from the graph which make cycles.

Node-weighted 2-edge-connected and k-edge-connected Steiner subgraphs

We consider a property $\sigma(H, K)$ which represents that H is a Steiner graph that contains all vertices in K and has at least 2 edge-disjoint paths between every pair of vertices. Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for all $v \in V$, and a subset of nodes $K \subseteq V$, a subgraph H for G and K that satisfies the property $\sigma(H, K)$ is termed as node-weighted 2-edge-connected Steiner subgraph. Likewise, we consider a property $\psi_k(H, K)$ which specifies that H is a graph that contains all vertices in K and has at least k edge-disjoint paths between every pair of terminals. Given an undirected graph $G = (V, E)$ with nonnegative weights w_v

for all $v \in V$, and a subset of nodes $K \subseteq V$, a subgraph H for G and K that satisfies the property $\psi_k(H, K)$ is referred to as node-weighted k-edge-connected Steiner subgraph.

Node-weighted 2-vertex-connected Steiner subgraphs

A property $\varsigma(H, K)$ is considered which represents that H is a graph that contains all vertices in K and has at least 2 vertex-disjoint paths between every pair of vertices. Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for all $v \in V$, and a subset of nodes $K \subseteq V$, a subgraph H for G and K satisfying the property $\varsigma(H, K)$ is referred to as node-weighted 2-vertex-connected Steiner subgraph.

Before we present an algorithm for the node-weighted δ-Steiner subgraph problem, and analyze its approximation ratio, we remark that for all the variants of the δ-Steiner problem considered in this thesis, it is easy to check whether a given instance of the problem with graph $G = (V, E)$ and set K of terminals admits a feasible solution. For this, one only needs to check whether there are at least the specified number of edge-disjoint or vertex-disjoint paths in G between any pair of vertices in K. This can be done in polynomial time using standard network flow techniques (see, e.g., [1]). Therefore, we can always assume that the given instance of the δ-Steiner subgraph problem admits at least one feasible solution.

In the following section, we present an algorithm for the node-weighted δ-Steiner subgraph problem.

4.2 Algorithm Description

Since we assume $w_u = 0$ for all $u \in K$, therefore, $c(u, v) = 0$ if $u, v \in K$. Our algorithm computes a δ-Steiner subgraph H using ρ-approximation algorithm. If

Algorithm 1 Algorithm for the node-weighted δ-Steiner subgraph problem

Input: An instance of the node-weighted δ-Steiner subgraph problem given by $G = (V, E)$ with nonnegative weights w_v for all $v \in V$ and a set $K \subseteq V$ of terminals.
Output: δ-Steiner subgraph H.

1 Assign weight $c(u, v)$ to every edge $(u, v) \in E$: $c(u, v) = w_u + w_v$
2 Compute a δ-Steiner subgraph H for graph G and terminals K with respect to the edge weights c using ρ-approximation algorithm.

H contains a Steiner node of degree 1, we repeatedly delete such Steiner nodes until no such Steiner node is left. Then the algorithm outputs H as solution to the given instance of the node-weighted δ-Steiner subgraph problem. The running time of Algorithm 1 is dominated by the time algorithm A takes in Step 2 to compute δ-Steiner subgraph H with respect to edge weights.

4.3 Analysis of Approximation Ratio

We consider graph classes Υ that are hereditary, i.e., if a graph is in the class, then any induced subgraph of that graph is also in the class. Furthermore, Υ has the following property P_d^δ, for some constant d:

(P_d^δ) Any graph H and any subset of vertices $K \subseteq V(H)$ in the class Υ satisfying the property $\delta(H, K)$ has a subgraph H' of maximum vertex degree d satisfying $\delta(H', K)$.

Let the graph $G = (V, E)$ of the given instance of the node-weighted δ-Steiner subgraph problem belong to a class of graphs with the above-stated property. Consider an optimal solution to this instance. The weight of the Steiner nodes in this solution is denoted by OPT_V. Let us denote the minimum total edge weight (with respect to edge weights c) of a δ-Steiner subgraph for G and terminal set K by

OPT_E. As our algorithm computes H by running a ρ-approximation algorithm for the edge-weighted version of the problem, the total edge weight of H, denoted by $c(H)$, thus satisfies $c(H) \leq \rho \cdot OPT_E$. Let the total node weight of H be $w(H)$. We analyze the approximation ratio by relating OPT_E to OPT_V and $c(H)$ to $w(H)$.

Lemma 2. *For classes of graphs satisfying (P_d^δ), it holds that* $OPT_E \leq d \cdot OPT_V$

Proof. Let H^* be the optimal solution to the node-weighted δ-Steiner subgraph problem. As in the example illustrated by Fig. 4.3, H^* may not belong to the same class of graphs to which G belongs. Therefore, we consider the subgraph of G induced by $V(H^*)$ and denote it by H'. Since property δ is monotone, this allows us to add edges to H^* in order to obtain H', and thus, H' satisfies property $\delta(H', K)$. By (P_d^δ), H' contains a subgraph H_d of maximum node degree d that satisfies $\delta(H_d, K)$. The total node weight of H_d is equal to that of H', as the set of nodes is the same. The total edge weight of H_d is at most $d \cdot OPT_V$, because the weight of each edge is the sum of the weights of its two end vertices and each vertex contributes to the weight of at most d edges. Hence, H_d is a feasible solution of edge weight at most $d \cdot OPT_V$, and the optimal solution with respect to edge weights cannot have larger weight. □

Lemma 3. $w(H) \leq c(H)/2$

Proof. As every Steiner node of H has degree at least two in H and $c(uv) = w_u + w_v$ for all edges uv, we have that every Steiner node contributes its weight once to the sum of all node weights but at least twice to the sum of all edge weights. Hence, $c(H) \geq 2w(H)$. □

Theorem 4.1. *Let δ be monotone. Then for hereditary classes of graphs satisfying (P_d^δ), Algorithm 1 is a* $0.5d\rho$*-approximation algorithm for the node-weighted δ-Steiner subgraph problem.*

Proof. Combining the previous lemmas, we get that $w(H) \leq c(H)/2 \leq 0.5\rho OPT_E \leq 0.5d\rho OPT_V$. □

Conclusion

Considering that nodes in a wireless ad-hoc networks can have different amount of available resources or willingness to participate in communication, we have allowed arbitrary node-weights to model the situation in graphs. Since terminals are part of every solution, we assume that they have weight zero, and the weight of the Steiner nodes in the solution H is minimized. Minimizing the weight of the Steiner nodes is a natural objective in this setting because it is desirable to minimize the cost of the nodes that forward packets but do not benefit from the multicast transmission themselves. The approximation ratio achieved is $0.5d\rho$, where d is the maximum degree bound of the solution, and ρ is the approximation ratio of the edge-weighted version of the problem. This result will be used in the analysis of our algorithms in remaining chapters.

Chapter 5

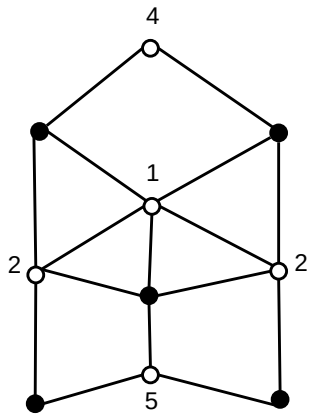
Node-Weighted Minimum Steiner Tree Problem

We discussed in Chapter 4 that there are properties δ which model the connectivity requirements in different types of Steiner subgraphs. Here, we consider the property $\mu(H, K)$ which specifies that H is a graph that contains all vertices in K and there is a path between every pair of vertices in K.

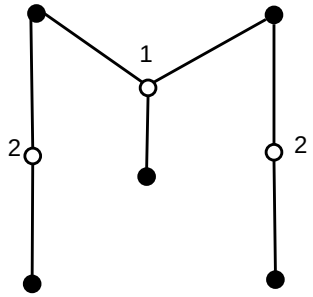
The node-weighted minimum Steiner tree problem is defined as follows:

Problem 2. *Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for $v \in V$, and a subset of nodes $K \subseteq V$ called* terminals*, compute a* Steiner subgraph *H for G and K that satisfies the property $\mu(H, K)$. The objective is to minimize the total weight of the vertices of H.*

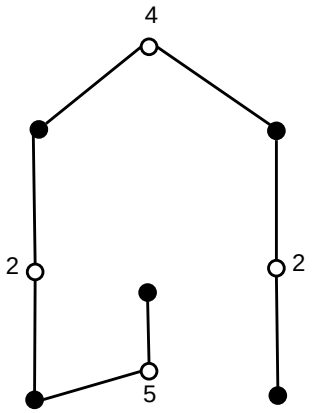
We can assume that the solution is a tree. If it is not a tree, we can just remove those edges without which the terminals are still connected in H. By deleting such edges we obtain a subgraph which is a tree. We can assume without loss of generality that the terminals have weight 0 as they are present in any solution and their weight increases the objective value of any solution by the same amount. So our goal is to minimize the total weight of the Steiner nodes of H. See Fig. 5.1 for



(a) An instance of the node-weighted Steiner tree problem denoted by G.



(b) A good solution for G with total node-weight 5.



(c) A bad solution for G with total node-weight 13.

Figure 5.1: Black nodes are terminals, white nodes are Steiner nodes.

an example of good and bad solutions for a given instance of the node-weighted minimum Steiner tree problem.

We assume that all leaves of a Steiner tree are terminals, so that all Steiner nodes are internal nodes with degree at least two. This assumption can be made because Steiner nodes of degree one can simply be removed from a Steiner tree. Furthermore, we consider only classes of graphs that are hereditary. For some constant d, we have a property (P_d^μ) for a class of graphs Υ, as follows:

$(\mathbf{P}_d^\mu)$ Any graph H in the class Υ, and a subset of vertices $K \subseteq V(H)$ satisfying the property $\mu(H, K)$ has a subgraph H' of maximum degree at most d satisfying $\mu(H', K)$.

We say that a class of graphs Υ has spanning tree degree bound d, if every connected graph H in Υ has a spanning tree with maximum degree at most d. It is clear that Υ satisfies the property (P_d^μ) if any connected graph H in Υ has a spanning tree of maximum degree at most d. Thus, we first derive results for classes of graphs with constant spanning-tree degree bound, and then for specific classes of graphs as corollaries.

As $NWMST$ is NP-hard for unit disk graphs and thus also for the more general graph classes we consider, we are interested in algorithms that compute solutions in polynomial time with good approximation ratios. In the following section the algorithm for $NWMST$ problem is presented.

5.1 Algorithm for Minimizing the Weight of Steiner Nodes

Algorithm 1 in Section 4.2 can be applied to the node-weighted minimum Steiner tree problem as follows:

Algorithm 2 Algorithm for $NWMST$

Input: An instance of $NWMST$ given by $G = (V, E)$ with nonnegative weights w_v for $v \in V$ and a set $K \subseteq V$ of terminals.
Output: Steiner tree H.

1 Assign weight $c(u, v)$ to every edge $(u, v) \in E$: $c(u, v) = w_u + w_v$
2 Compute a subgraph H for G satisfying $\mu(H, K)$ with respect to edge weights using the algorithm by Byrka et al [8].

We assume that $w_u = 0$ for all $u \in K$, thus $c(u, v) = 0$ if $u, v \in K$.

Our algorithm computes a Steiner tree H for graph G and terminals K with respect to the edge weights c using the algorithm of Byrka et al. [8]. Then the algorithm outputs the tree H as solution to the given instance of $NWMST$.

We already know that Theorem 4.1 can be applied to a subgraph H if δ is monotone, i.e., adding an edge to a Steiner subgraph H and deleting a Steiner node of degree 1 from it does not destroy the property. This property of monotonicity does not hold if H is required to be a Steiner tree (because a Steiner tree will no more be a tree if we add an edge to it). Thus, the output of the algorithm, i.e., H, is considered as a subgraph that connects all terminals instead of a tree structure. This allows us to add edges to H in order to find its induced subgraph. Recall that we are interested in the induced subgraph of H for the purpose of analysis of the algorithm, to ensure that we are taking into account the maximum degree bound d of the subgraph which belongs to same class of graphs to which the input graph belongs.

Theorem 5.1. *For hereditary graph classes satisfying (P_d^μ), there is a $0.695d$-approximation algorithm for the Node-Weighted Minimum Steiner Tree problem.*

Proof. According to Theorem 4.1, for hereditary classes of graphs that satisfy P_d^δ, we have a $0.5d\rho$-approximation algorithm for the node-weighted δ-Steiner subgraph problem, where δ is monotone. Since we consider the property $\mu(H, K)$ which is

monotone, and represents that H is a Steiner subgraph in which all terminals are connected, Theorem 4.1 implies that we have $0.5d\rho$-approximation algorithm for the node-weighted minimum Steiner tree problem in classes of graphs that satisfy (P_d^{μ}). As the algorithm by Byrka et al. [8] is a 1.39-approximation algorithm for the edge-weighted Steiner tree problem, we have an approximation algorithm with ratio $0.5 \cdot 1.39d = 0.695d$ for the node-weighted minimum Steiner tree problem. □

5.1.1 *NWMST* in Special Graph Classes

In this section, we apply Theorem 5.1 to several classes of graphs that are frequently employed to model wireless ad-hoc networks.

Unit Disk Graphs

First, we obtain directly the result for unit disk graphs that is presented by Zou et al. in [62].

Corollary 1. *There is a* 3.475*-approximation algorithm for the Node-Weighted Minimum Steiner Tree problem in unit disk graphs (Zou et al. [62]).*

Proof. As shown by Wu et al. [59], any connected unit disk graph admits a spanning tree in which the degree of every vertex is at most five. By applying Theorem 5.1 with $d = 5$, we obtain approximation ratio $0.695 \cdot 5 = 3.475$. □

α-Unit Disk Graphs

Lemma 4. *For every* $0 < \alpha \leq 1$*, the class of* α*-unit disk graphs has spanning-tree degree bound at most* $6 + 8/\alpha^2 + 4\sqrt{2}/\alpha$*.*

Proof. Consider any connected α-unit disk graph $G = (V, E)$. Let the weight w_{uv} of an edge $uv \in E$ be the Euclidean distance between the positions of nodes u and

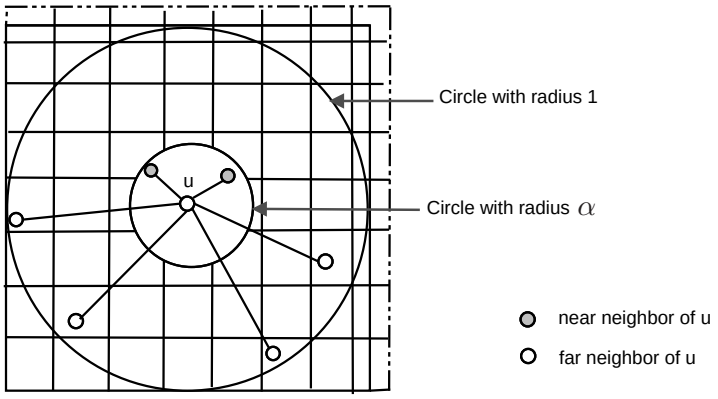


Figure 5.2: Neighbors of a node in an α-unit disk graph

v, and let T be a minimum spanning tree of G with respect to these edge weights. Consider an arbitrary node u of G. The degree of node u in T can be calculated as the number of neighbors that are at distance at most α from u plus the number of neighbors that have distance greater than α from u. We call the former type of neighbors the *near* neighbors, the latter type the *far* neighbors. We bound the number of neighbors of each type separately. First, consider the near neighbors of u in T. We can use the method of [59] to prove that T can be chosen so that there are at most 5 such neighbors.

Now consider the far neighbors of node u in T. All such neighbors lie in a ring around u that is bounded by circles of radius α and 1 with center u, see Fig. 5.2. This ring is contained in a square with side length 2 and center u. We cover this square using square *cells* of side length $\alpha/\sqrt{2}$ (and, thus, diameter α). We claim that each cell can contain at most one far neighbor of u in T. If a cell contained two far neighbors v and w of u in T, the weight of T could be decreased by removing one of the edges uv and uw (both of which have length more than α), and adding the edge vw (which has length at most α). Hence, the number of far neighbors of u is bounded by the number of cells needed to cover the ring, which is in turn bounded by the number of cells needed to cover a square of area 2×2. This number

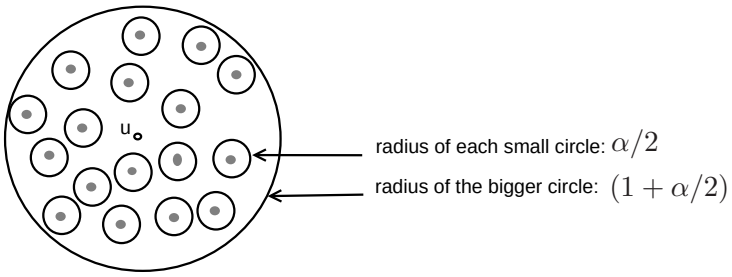


Figure 5.3: Independent neighbors of an arbitrary node u in an α-unit disk graph

of cells is at most $\left\lceil \frac{2}{\frac{\alpha}{\sqrt{2}}} \right\rceil \cdot \left\lceil \frac{2}{\frac{\alpha}{\sqrt{2}}} \right\rceil \leq (\frac{2\sqrt{2}}{\alpha}+1) \cdot (\frac{2\sqrt{2}}{\alpha}+1) = \frac{8}{\alpha^2} + \frac{4\sqrt{2}}{\alpha} + 1$.

Considering both types of neighbors of u, we get that u has degree at most $5 + 8/\alpha^2 + 4\sqrt{2}/\alpha + 1 = 6 + 8/\alpha^2 + 4\sqrt{2}/\alpha$ in T. As the argument can be applied to any node u, the claim follows. □

Corollary 2. *For α-unit disk graphs, there is a $0.695 \cdot (6 + 8/\alpha^2 + 4\sqrt{2}/\alpha)$-approximation algorithm for the $NWMST$ problem.*

This result also applies to disk graphs with bounded diameter ratio. A disk graph is a graph where each node corresponds to a disk in the plane and two nodes are adjacent if the corresponding disks have a non-empty intersection. The diameter ratio of a disk graph is the ratio between the largest and the smallest diameter of any disk in the graph. If a disk graph has diameter ratio at most D for some constant D, we can assume without loss of generality that the largest disk has diameter 1 and the smallest disk has diameter at least $1/D$. The graph is then an α-unit disk graph with $\alpha = 1/D$. Hence, Lemma 4 and Corollary 2 imply that disk graphs with diameter ratio at most D have spanning-tree degree bound $O(D^2)$ and admit an $O(D^2)$-approximation algorithm for the $NWMST$ problem.

We now show a bound on the maximum number of independent neighbors of a node in α-unit disk graphs. We will use this bound later to achieve a better result for $NWMST$ than what we obtained in Corollary 2 for α-unit disk graphs.

Lemma 5. *For every $0 < \alpha \leq 1$, the number of independent neighbors of every node in the class of α-unit disk graphs is at most $4/\alpha^2 + 4/\alpha + 1$.*

Proof. Consider any α-unit disk graph $G = (V, E)$. We bound the maximum number of independent neighbors of an arbitrary node u in G by considering a disk of radius $(1 + \alpha/2)$ with center u (see Fig 5.3). We draw a small disk of radius $\alpha/2$ around each independent neighbor of u. As all these small disks are disjoint (since independent neighbors of u must have distance larger than α) and contained in the disk of radius $(1 + \alpha/2)$, the number of these small disks is at most:

$$\frac{(1+\alpha/2)^2\pi}{(\alpha/2)^2\pi} = \frac{1+\alpha^2/4+\alpha}{\alpha^2/4} = 4/\alpha^2 + 4/\alpha + 1.$$

Thus, the number of independent neighbors of any node in an α-unit disk graph is at most $4/\alpha^2 + 4/\alpha + 1$. □

$(\lambda + 1)$-Claw-Free Graphs

Next, we consider the class of $(\lambda + 1)$-claw-free graphs. In $(\lambda + 1)$-claw-free graphs, the number of independent neighbors of every node is bounded by λ. First we show spanning tree degree bound in $(\lambda + 1)$-claw-free graphs, and then apply Theorem 4.1.

Lemma 6. *For every integer $\lambda \geq 1$, the class of $(\lambda + 1)$-claw-free graphs has spanning-tree degree bound at most $\lambda + 1$.*

Proof. We can prove that there exists a spanning tree of bounded maximum degree in $(\lambda+1)$-claw-free graphs by constructing a tree with depth first search (DFS) (see [12] for a description of DFS). Given a $(\lambda + 1)$-claw-free graph $G = (V, E)$, start with an arbitrary node and construct a DFS tree T. Consider an arbitrary node u and any two of its children denoted by v and w in T. If there was an edge v-w in G, then according to the construction of any DFS tree, v would have been either a descendant of w or an ancestor of w in T, but both of them could not

belong to a common parent. Therefore, the two cases that there is an edge v-w in the original graph G, and both v and w have the same parent in T cannot occur. As we have assumed that v and w have a common parent u, nodes v and w are independent nodes in the original graph. Thus, children of every node in a DFS tree are independent nodes. Furthermore, every node can have at most λ independent neighbors in a $(\lambda+1)$-claw-free graph, thus, there are at most λ children of every node in T. This means that the number of neighbors of u is at most $\lambda+1$, i.e., λ children plus one parent. Hence, it is proved that T has maximum degree bound at most $\lambda+1$. □

We also give an alternative proof for Lemma 6 as follows:

Let $G=(V,E)$ be any connected, $(\lambda+1)$-claw-free graph. We give an algorithm that constructs a spanning tree of G with maximum degree at most $\lambda+1$. The algorithm grows a subtree T of G, initially consisting of a single node, into a spanning tree by repeatedly picking a leaf node v and adding a maximal independent set of its neighbors outside T as children of v. More formally, the algorithm can be described as follows:

1. Let T be the subtree of G consisting of an arbitrary node u of G.
2. **while** T is not yet a spanning tree **do**
   (i) Pick any leaf v of T that has at least one neighbor outside T.
   (ii) Compute a maximal independent set I among the neighbors of v that are not in T.
   (iii) Add all nodes of I as children of v to T.
3. **return** T.

First, we show that the algorithm computes a spanning tree. The algorithm could only fail to compute a spanning tree if it reaches a situation in which T is not

yet a spanning tree, but in Step 2(i) there is no leaf of T with a neighbor outside T. Assume that this happens. Let U be the set of nodes in T and W the set of nodes outside T. Since G is connected, there must be a node $u \in U$ and a node $w \in W$ such that $uw \in E$. The node u cannot be a leaf of T, because we assume that no leaf has an edge to a node in W. Hence, u is an internal node of T. We can assume that u is chosen to be the deepest node in T that is adjacent to w. In an earlier step of the algorithm, when the current tree was a subtree T' of T and u was a leaf of T', the algorithm has added a maximal independent set I of u's neighbors as children of u to the tree. As w was not added to I, w must be adjacent to a node in I, and hence w must be adjacent to a child of w. This is impossible because we have assumed that u is the deepest node in T that is adjacent to w. We have reached a contradiction, and hence the situation where the algorithm fails to compute a spanning tree cannot occur.

Now we analyze the degree of the constructed spanning tree. As G is $(\lambda+1)$-claw-free, the independent set I computed in Step 2(ii) has cardinality at most λ. Therefore, every node of T has at most λ children and at most one parent. Consequently, the degree of every node of the tree is at most $\lambda+1$.

Corollary 3. *There is a* $0.695 \cdot (\lambda+1)$*-approximation algorithm for the* $NWMST$ *problem in* $(\lambda+1)$*-claw-free graphs.*

α-unit disk graphs are $(4/\alpha^2 + 4/\alpha + 2)$-claw-free due to Lemma 5. Therefore, Corollary 3 implies that we have an algorithm with approximation ratio $0.695 \cdot (4/\alpha^2 + 4/\alpha + 2) = (2.78/\alpha^2 + 2.78/\alpha + 1.39)$ for the $NWMST$ problem in α-unit disk graphs. Thus, due to Corollary 3 we obtain a better result for the $NWMST$ problem than the result obtained in Corollary 2 for α-unit-disk graphs. However, for unit-disk graphs, Corollary 3 does not give us better result than what we have obtained in Corollary 1.

Graph Classes	Result obtained by applying Theorem 5.1 directly	Result implied by Corollary 3
unit disk graphs	3.475	4.71
α-unit disk graphs	$(4.17 + 5.56/\alpha^2 + 2.78\sqrt{2}/\alpha)$	$(2.78/\alpha^2 + 2.78/\alpha + 1.39)$
bounded independence graphs	$--$	$0.695 \cdot (\lambda + 1)$
bi-directional disk graphs	$--$	$(12.51 \lceil \log_2 m \rceil + 9.035)$, for $m > 1$

Table 5.1: Results obtained for the node-weighted Steiner tree problem in restricted graph classes.

Bounded Independence Graphs

Recall from Chapter 2 that in bounded independence graphs, the number of independent neighbors of every node that are r hops away is bounded by a polynomial in r, i.e., $p(r)$. For $\lambda = p(1)$, bounded independence graphs are $(\lambda + 1)$-claw-free. Thus, Corollary 3 implies the following result:

Corollary 4. *There is a* $0.695 \cdot (\lambda + 1)$*-approximation algorithm for the* $NWMST$ *problem in bounded independence graphs.*

Bi-Directional Disk Graphs

In bi-directional disk graphs, every node has at most $6(3 \lceil \log_2 m \rceil + 2)$ independent neighbors, for $m > 1$ [53]. Recall that $m = r_{max}/r_{min}$, where r_{max} is the maximum transmission range and r_{min} is the minimum transmission range. Therefore, bi-directional disk graphs are $(18 \lceil \log_2 m \rceil + 13)$-claw-free, for $m > 1$. For $m = 1$, a bi-directional disk graphs is a unit-disk graph and we have 3.475 approximation ratio for $NWMST$ in unit disk graphs due to Corollary 1. For $m > 1$, we get the following result for bi-directional disk graphs.

Corollary 5. *There is a* $0.695 \cdot (18 \lceil \log_2 m \rceil + 13) = (12.51 \lceil \log_2 m \rceil + 9.035)$*-approximation algorithm for the NWMST problem in bi-directional disk graphs, for* $m > 1$.

Conclusion

We have shown that by defining suitable edge weights and then applying an algorithm with best known approximation ratio for the edge-weighted Steiner tree problem, we achieve a good approximation ratio for the node-weighted minimum Steiner tree problem in restricted graph classes used to model wireless ad-hoc networks. An overview of the achieved approximation results is given in Table 5.1.

Chapter 6

Node-Weighted 2-Edge- and 2-Vertex-Connected Steiner Subgraphs

Multicast communication in wireless ad-hoc networks can be achieved with Steiner trees, however, due to the lack of fixed infrastructure and wireless medium, these networks are prone to link and node failures. Sender and receivers may disconnect even if a single link or node fails in the tree. Thus, trees are not reliable for communication in such networks, and in scenarios where fault-tolerance is a key requirement, one looks at communication structures that offer higher connectivity, so that in the event of link or node failure an alternative path could be used to transmit packets. In this chapter, we study the 2-edge-connected Steiner subgraph problem, the 2-vertex-connected Steiner subgraph problem, and the 2-edge-connected group Steiner subgraph problem with respect to node weights in restricted graph classes that model wireless ad-hoc networks. Our algorithm yields better approximation ratio in these graphs than the previous results in arbitrary graphs for these problems.

6.1 Node-Weighted 2-Edge-Connected Steiner Subgraph Problem

We consider the property $\sigma(H, K)$ representing that H is a graph that contains all vertices in K and has at least 2 edge disjoint paths between every pair of vertices. We define the node-weighted 2-edge-connected Steiner subgraph problem ($NW2ECS$) as follows:

Problem 3. *Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for $v \in V$, and a subset of nodes $K \subseteq V$ (*terminals*), compute a subgraph H for G and K that satisfies the property $\sigma(H, K)$. The objective of the $NW2ECS$ problem is to minimize the total weight of the vertices of H.*

Unlike Steiner trees, 2-edge-connected Steiner subgraphs exhibit some level of fault tolerance due to 2 edge-disjoint paths between every pair of vertices. Note that the alternative requirement of 2-edge-disjoint paths only between every pair of terminals does not change the problem.

$\sigma(H, K)$ satisfies the property of monotonicity (see Chapter 4 for a definition of monotonicity). We can assume that H does not contain any Steiner nodes of degree less than 2, because such vertices could be removed from the solution while reducing the cost and maintaining feasibility. In fact, a 2-edge-connected graph cannot contain vertices of degree less than 2. Furthermore, addition of edges to H does not harm the property of being a 2-edge-connected Steiner subgraph. Therefore, we can add edges to H for the analysis of the algorithm.

6.1.1 Computing 2-Edge-Connected Steiner Subgraphs

The following is the algorithm for the node-weighted 2-edge-connected Steiner subgraph problem.

Algorithm 3 Algorithm for $NW2ECS$

Input: An instance of $NW2ECS$ given by $G = (V, E)$ with nonnegative node weights w_v for $v \in V$ and a set $K \subseteq V$ of terminals.
Output: 2-edge-connected Steiner subgraph H.

1 Assign weight $c(u, v)$ to every edge $(u, v) \in E$: $c(u, v) = w_u + w_v$
2 Compute a subgraph H for G satisfying $\sigma(H, K)$ with respect to edge weights using the algorithm by Jain [26].

Note that $c(u, v) = 0$ if $u, v \in K$. As the edge-weighted 2-edge-connected Steiner subgraph problem is a special case of the generalized Steiner network problem, we use the algorithm by Jain [26] which is the best known approximation algorithm for the generalized Steiner network problem, and also for its special case, the 2-edge-connected Steiner subgraph problem with respect to edge weights.

It is easy to check whether a given instance of the problem $NW2ECS$ wit graph $G = (V, E)$ and set K of terminals admits a feasible solution. For this, one is required to check whether there are at least two edge-disjoint paths in G between every pair of vertices in K. This can be done in polynomial time using standard network flow techniques [1]. Therefore, we assume that the given instance of the problem $NW2ECS$ has at least one feasible solution.

6.1.2 Analysis of Approximation Ratio

First, we analyze the algorithm for graph classes that have the following property, for some constant d:

($\mathbf{P}_d^\sigma$) Any 2-edge-connected graph in the class contains a spanning 2-edge-connected subgraph of maximum vertex degree d.

In subsequent sections, we will then show that UDG and quasi-UDG satisfy this property.

Assume that the graph $G = (V, E)$ of the given instance of $NW2ECS$ belongs to a class of graphs with property (P_d^σ). We analyze the approximation ratio using an adaptation of analogous results for the node-weighted δ-Steiner subgraph problem studied in Chapter 4.

Theorem 6.1. *For hereditary classes of graphs satisfying (P_d^σ), Algorithm 3 is a d-approximation algorithm for the problem $NW2ECS$.*

Proof. By Theorem 4.1, Algorithm 1 (see Section 4.2) is a $0.5d\rho$-approximation algorithm for the node-weighted δ-Steiner subgraph problem in hereditary classes of graphs that satisfy (P_d^δ), where δ is monotone. As our considered property $\sigma(H, K)$ is monotone, Theorem 4.1 implies that we have a $0.5d\rho$-approximation algorithm for the node-weighted 2-edge-connected Steiner subgraph problem in classes of graphs that satisfy (P_d^σ). Since Jain's algorithm has approximation ratio 2, Algorithm 3 has approximation ratio $0.5d \cdot 2 = d$ for the node-weighted 2-edge-connected Steiner subgraph problem. □

Before we consider UDG and quasi-UDG, we state the following auxiliary lemma that follows directly from Menger's theorem.

Lemma 7. *If there are at least two edge-disjoint paths between u and v, and at least two edge-disjoint paths between v and w, then there are also at least two edge-disjoint paths between u and w.*

Proof. Recall that by Menger's theorem (see, e.g., [1]), for any pair of distinct nodes x and y, the size (number of edges) of the minimum cut between x and y is equal to the maximum number of edge-disjoint paths between x and y. Consider any cut C that separates u and w. The node v is connected to each of the nodes u and w via two edge-disjoint paths.

- Case 1: C separates u and v, i.e., v is located on the side of C where w is located. We know that there are at least two edge-disjoint paths between u

and v. So, there are at least two edges going across C, which means that the size of C is at least 2.

- Case 2: v is on the side of C where u lies. There are two disjoint paths between v and w, and C is a cut separating these two nodes. Again, the size of C is at least 2.

In either case, there are at least two edges in the cut C. As the choice of C was arbitrary, we have that any cut that separates u and w has size at least 2. By Menger's theorem, this implies that there are at least 2 edge-disjoint paths between u and w. □

If there are two edge-disjoint paths between two nodes u and v, we also say that u and v *are in a cycle*.

6.1.3 Unit Disk Graphs

In this section, we find a 2-edge-connected spanning subgraph of maximum degree at most 12 in unit disk graphs, and then apply Theorem 6.1 to achieve approximation ratio 12 for $NW2ECS$.

Lemma 8. *Every* 2*-edge-connected unit disk graph has a* 2*-edge connected spanning subgraph of maximum degree at most* 12*.*

Proof. Let $G = (V, E)$ be a 2-edge-connected unit disk graph. We show how to construct a 2-edge-connected spanning subgraph G' of small degree. First, we determine a minimum spanning tree T (with respect to edge weights that represent the Euclidean distance between the endpoints of the edge). Then, we add edges to T so that it becomes 2-edge-connected. The construction is as follows:

1. Compute a minimum spanning tree T with respect to edge weights given by Euclidean distance, and initialize $G' = T$.

2. For all edges $e \in E$ that are not in T, in order of non-decreasing length:

3. if $G' \cup \{e\}$ puts at least one bridge edge from G' into a cycle, then add e to G'.

Here, an edge of G' is a bridge edge (or simply a bridge) if it is not contained in any cycle and thus its removal disconnects G'. If a bridge edge in G' is contained in a cycle in $G' \cup \{e\}$ it is no longer a bridge edge.

We observe that after the construction has considered an edge uv in the for-loop, either it has added the edge or there were already two edge-disjoint paths between u and v in G'. In either case, we have that after an edge uv has been processed, u and v are in a cycle.

It is easy to see that if G is 2-edge-connected, the graph G' produced by the construction is 2-edge-connected. It remains to analyze the maximum node degree of the final G'. Consider an arbitrary node u. We divide the surrounding area of u into six equal 60° sectors. We choose the sectors in such a way that none of the neighbors of u in G lies on the boundary between two sectors. Note that each sector contains at most one node that is a neighbor of u in the spanning tree T. (If there were two neighbors of u in the same sector, one of the two edges joining u and these neighbors could be replaced by the shorter edge between these two neighbors to give a spanning tree of smaller cost, a contradiction.) The degree of u in the final G' is the sum of the number of adjacent nodes of u in all six sectors. We claim that u has at most 2 neighbors in each sector in G'. To prove this, consider the following cases.

Case 1: Consider a 60° sector of u that does not contain a spanning tree neighbor of u. Assume that there are two nodes x and z in the sector such that the construction has added edges between u and these two nodes to G'. See Figure 6.1(a). Let y be a third node in the sector that the construction considers. If

xy and yz are in T, then $uxyzu$ is a cycle in G' containing y and u, so yu is not added.

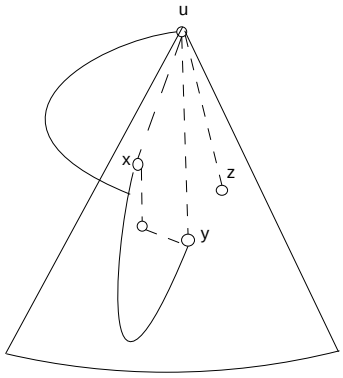
Now assume that at least one of the two edges (xy and yz) is not in T. Let us assume xy is not in T. (The case that yz is not in T is analogous.) The construction adds edges in order of increasing length, so xy is considered before yu. This means that y and x are already in a cycle when yu is considered. Furthermore, x and u are in a cycle because the construction has added the edge xu and there is also a path connecting x and u in T. By Lemma 7, if u and x are in a cycle and x and y are in a cycle, then u and y are also in a cycle. Therefore, the construction does not add y as third neighbor of u.

Case 2: u has one neighbor v in the sector such that $uv \in T$. Assume that the construction has added x as a second neighbor of u in the sector. Note that the distance from u to x is at least as large as the distance from u to v, because otherwise the edge vu would not be in any minimum spanning tree. (It would be the longest edge in the cycle $uvxu$.) Assume z is another node in the sector and the construction considers the edge zu.

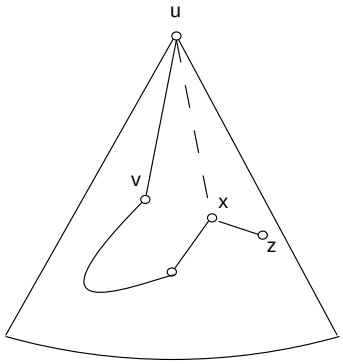
(i) If zv and zx are in T: Then $zvuxz$ is a cycle. So, the construction does not add zu.

(ii) If zx is not in T (and zv may or may not be in T): Before considering zu, the construction has considered zx. From then on, z and x are in a cycle. Furthermore, x and u are in a cycle. Therefore, by Lemma 7, u and z are in a cycle, and the edge zu is not added by the construction.

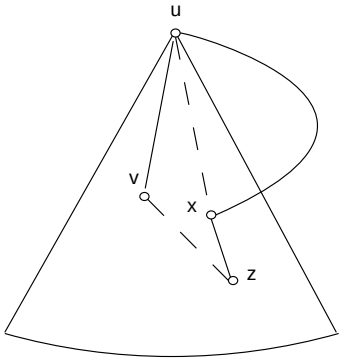
(iii) If zx is in T and zv is not in T: There can be two different ways in which x is connected to u, v via a path in T. (Furthermore, if z lies on that path then z and u are in a cycle and zu does not get added to G' by the construction.



(a) Case 1



(b) Case 2(iii)(a)



(c) Case 2(iii)(b)

Figure 6.1: Illustration of cases in the proof of Lemma 8. Edges in T are drawn solid, other edges dashed.

Hence, we assume in the following that z does not lie on the path in T between x and u, v.)

(a) The path in T from x to u, v reaches v before u. See Figure 6.1(b). Observe that u and v are in a cycle as ux has been added to G'. Moreover, the construction considers vz before uz is considered. Therefore, v and z are already in a cycle when the construction considers uz. By Lemma 7, if u and v are in a cycle and v and z are in a cycle, then u and z are also in a cycle. Thus, the construction does not add zu.

(b) The path in T from x to u, v reaches u before v. See Figure 6.1(c). vz is considered before uz. If the construction adds vz, then u and z lie on the cycle consisting of the edge vz and the path in T between v and z, so uz does not get added. If vz is not added, this shows that uv is not a bridge. (If uv was a bridge, the addition of vz would have put it into a cycle, and hence the construction would have added vz.) Hence, v and z are in a cycle, and u and v are in a cycle. By Lemma 7, u and z are in a cycle, and the construction does not add uz.

We have shown that in each of the six sectors, it is impossible that the construction adds an edge to a node that would become a third neighbor to u in that sector. Therefore, the final G' is a 2-edge-connected spanning subgraph in which each node has at most 12 neighbors. □

By applying Theorem 6.1 with $d = 12$, we obtain the following corollary.

Corollary 6. *There is a* 12*-approximation algorithm for NW2ECS in unit disk graphs.*

6.1.4 α-Unit Disk Graphs

In this section, we find a 2-edge-connected spanning subgraph of maximum degree at most $14 + 16/\alpha^2 + 8\sqrt{2}/\alpha$ in α-UDG. When Theorem 6.1 is applied with $d = 14+16/\alpha^2+8\sqrt{2}/\alpha$, we get approximation ratio $14+16/\alpha^2+8\sqrt{2}/\alpha$ for $NW2ECS$.

Lemma 9. *For every $0 < \alpha \leq 1$, every 2-edge-connected α-UDG has a 2-edge-connected spanning subgraph of node degree at most $14 + 16/\alpha^2 + 8\sqrt{2}/\alpha$.*

Proof. Apply the construction in the proof of Lemma 8 to produce a 2-edge-connected spanning subgraph G'. To analyze the neighborhood of a node in an α-UDG, we use the same framework as in Chapter 5. The neighbors of a node u are classified as *near* (within distance at most α from u) and *far* (distance greater than α) neighbors. The proof of Lemma 8 can be used to show that u has at most 12 near neighbors in the final G'. In the remainder of the proof, we consider only far neighbors. To bound the number of far neighbors, we cover a 2×2 square containing all neighbors of u using squares of side length $\alpha/\sqrt{2}$ called *cells*, see Figure 6.2. Note that any two nodes in the same cell have distance at most α and hence are adjacent. We show that the number of far neighbors of u in the final G' is at most twice the number of cells. After computing a minimum spanning tree T, there is at most one neighbor of u in every cell. We show that if a cell contains a neighbor of u in T, then at most one more node in that cell can be added as a neighbor of u by our construction, and otherwise, at most two neighbors can be added to u.

Consider three nodes v, w, x in a cell and assume that in the minimum spanning tree, there is no neighbor of u in this cell. Assume that v and w have already been added as neighbors of u in this cell and we are now considering the edge ux. We claim that the construction would not add x as third neighbor of u. To show this, we can use arguments analogous to Case 1 of the proof of Lemma 8: If vx and wx

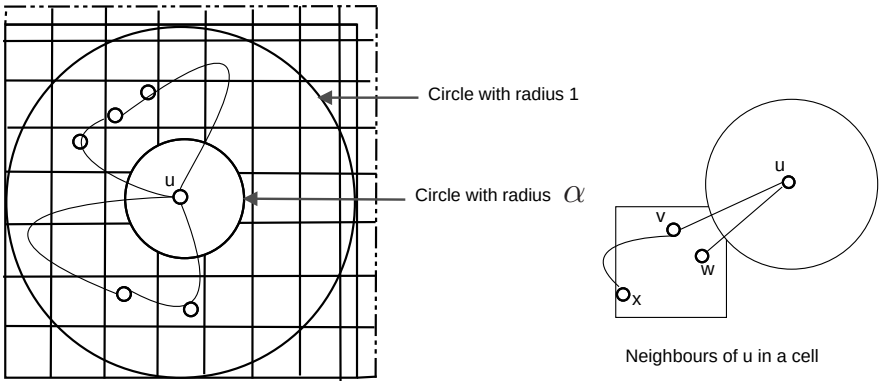


Figure 6.2: Neighbors of u in an α-unit disk graph

are both in T, then we have a cycle $uvxwu$ and therefore, xu does not get added to T. If at least one of vx and wx is not in T, assume that vx is not in T. Then the algorithm considers vx before xu is considered, so by the time xu is considered v and x are already in a cycle. By Lemma 7, if u and v are in a cycle and v and x are in a cycle, then u and x are also in a cycle. Therefore, the construction does not add x as third neighbor of u.

Now consider the case that u has one neighbor v' in T in the cell, a second node w' has been added as neighbor of u in this cell, and we are now considering the edge ux' for a third node x' in this cell. Here, the same arguments apply as we have described in Case 2 of the proof of Lemma 8, and it again follows that ux' is not added.

Thus, in every cell, u has at most two far neighbors in the end. Therefore, the number of far neighbors of u is at most twice the number of cells, and the number of cells is at most

$$\left\lceil \frac{2}{\frac{\alpha}{\sqrt{2}}} \right\rceil \cdot \left\lceil \frac{2}{\frac{\alpha}{\sqrt{2}}} \right\rceil \leq (\frac{2\sqrt{2}}{\alpha} + 1) \cdot (\frac{2\sqrt{2}}{\alpha} + 1) = \frac{8}{\alpha^2} + \frac{4\sqrt{2}}{\alpha} + 1\,.$$

The total number of *near* and *far* neighbors of u is at most $12+16/\alpha^2+8\sqrt{2}/\alpha+2 = 14 + 16/\alpha^2 + 8\sqrt{2}/\alpha$. □

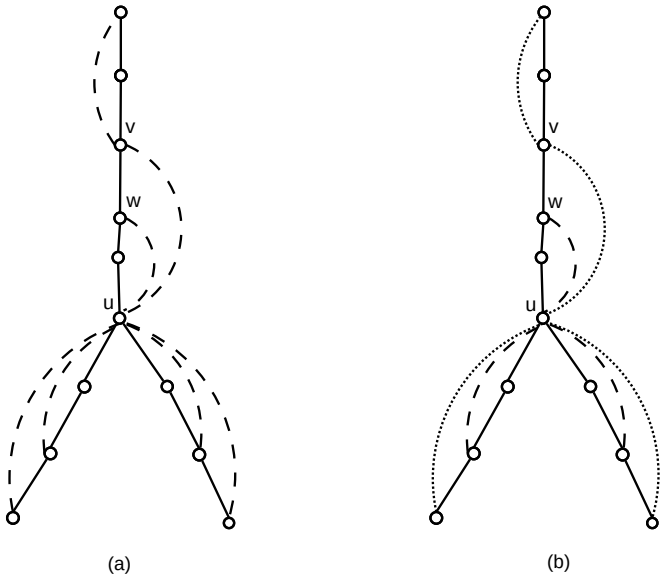


Figure 6.3: solid lines are edges of DFS tree, dashed lines are back edges in DFS, and dotted lines in (b) are edges chosen by our construction for 2-edge-connectivity in a $(\lambda+1)$-claw-free graph.

By applying Theorem 6.1 with $d = 14+16/\alpha^2+8\sqrt{2}/\alpha$ we obtain the following corollary.

Corollary 7. *There is an approximation algorithm with approximation ratio* $14 + 16/\alpha^2 + 8\sqrt{2}/\alpha = O(1/\alpha^2)$ *for* $NW2ECS$ *in* α*-unit disk graphs.*

6.1.5 $(\lambda+1)$-Claw Free Graphs

We construct a 2-edge-connected spanning subgraph of maximum degree bound d in $(\lambda+1)$-claw free graphs.

Lemma 10. *For every* $\lambda \geq 1$*, every* 2*-edge-connected* $(\lambda+1)$*-claw-free graph has a* 2*-edge-connected spanning subgraph of maximum degree at most* $2(\lambda+1)$*.*

Proof. Let $G = (V, E)$ be a 2-edge-connected $(\lambda+1)$-claw-free graph. In order to construct a 2-edge-connected spanning subgraph G' of bounded degree, we first determine a spanning tree T, and then, we add edges to T so that it becomes a 2-edge-connected subgraph. The construction is as follows:

```
1 Compute a DFS tree T for graph G = (V, E)
2 G' = T
3 Process tree T top-down: for each node v ∈ V
4     for all children c of v in T
5         if v-c is a bridge in G' then
6             add to G' an edge from v or a node above v to a node in the
              subtree below c, with deepest lower end-point.
```

Each node can have at most λ children in the DFS tree T. We claim that when we add edges to T to construct a 2-edge-connected spanning subgraph G', each node gets at most $(\lambda + 1)$ new neighbors. Therefore, the degree of each node is at most $2(\lambda + 1)$ in G'.

We show how the degree of every node in T increases by at most $(\lambda + 1)$ while computing G'. Consider an arbitrary node u. Let us first consider the maximum number of neighbors added to u from above. Assume that the algorithm adds an edge from above with lower end point u. Consider the first such edge. Assume that the edge is added when the algorithm considers the bridge v-w, where v is the parent of w in T. Let t be the upper end point of this added edge. t can be equal to v or a node above v. After the addition of this edge, all edges on the path from v to u in T cannot be bridges. Therefore, it is impossible that the algorithm adds a second edge from above to u.

Let us now check how many neighbors can be added to u from below in G'. Children of u in T are independent nodes, and we know that in a $(\lambda + 1)$-claw-free graph, every node can have at most λ independent neighbors, therefore, u has at most λ children in T. This means that u has λ children in G' initially. We show

that every node can get at most λ new neighbors from below in G' while adding edges for 2-edge-connectivity. As edges are added to deepest descendants, none of the lower end points of the edges being added to u from below is an ancestor of another. Hence, all new neighbors of u from below are independent nodes. The case that more than λ new neighbors are added to u, therefore, cannot occur.

Now, the degree of u in G' is the sum of the number of neighbors of u in T and the number of neighbors added to u from above (ancestors) and below (descendants) in G'. Hence, u has at most $2(\lambda + 1)$ neighbors in the 2-edge-connected spanning subgraph G'. □

By applying Theorem 6.1 with $d = 2(\lambda + 1)$, we obtain the following corollary.

Corollary 8. *There is an approximation algorithm with approximation ratio* $2(\lambda + 1)$ *for NW2ECS in* $(\lambda + 1)$*-claw-free graphs.*

We know that unit disk graphs are 6-claw-free [59], therefore, the above corollary implies a 12-approximation algorithm for unit disk graphs which is the same result as obtained in Corollary 6. As α-unit disk graphs are $(4/\alpha^2 + 4/\alpha + 2)$-claw-free due to Lemma 5, Corollary 8 implies that we have an algorithm with approximation ratio $2 \cdot (4/\alpha^2 + 4/\alpha + 2)$ which is better than the approximation ratio $14 + 16/\alpha^2 + 8\sqrt{2}/\alpha$ obtained in Corollary 7. An overview of the achieved approximation results is given in Table 6.1.

6.2 Node-Weighted 2-Edge-Connected Group Steiner Subgraph Problem

Given a graph $G = (V, E)$ with edge-costs c_e for all $e \in E$, a root node $r \in V$, and a collection of M subsets of nodes $K = \{K_1, ..., K_M\}$ of $V \setminus \{r\}$. The edge-weighted

Graph Classes	Result obtained by applying Theorem 6.1 directly	Result implied by Corollary 8
unit disk graphs	12	12
α-unit disk graphs	$14 + 16/\alpha^2 + 8\sqrt{2}/\alpha$	$2 \cdot (4/\alpha^2 + 4/\alpha + 2)$
bounded independence graphs	$--$	$2 \cdot (\lambda + 1)$
bi-directional disk graphs	$--$	$2 \cdot (18 \lceil \log_2 m \rceil + 13)$, for $m > 1$

Table 6.1: Results obtained for the node-weighted 2-edge-connected Steiner subgraph problem in restricted graph classes.

fault-tolerant group Steiner problem asks to compute a subgraph H of minimum total edge-weight that contains at least 2 edge- or vertex-disjoint paths from each group K_i to r. Khandekar et al. [29] studied various versions of this problem. We consider the node-weighted version of the problem termed as $EC\text{-}FTGS$-2 in [29].

We consider the property $\chi(H, K)$ which represents that H is a graph that contains at least 2 edge-disjoint paths from each group K_i of terminals to a root vertex r and the end-points of these paths in K_i are distinct. The node-weighted version of the problem $EC\text{-}FTGS$-2 is defined as follows:

Problem 4. *Given a graph $G = (V, E)$ with nonnegative node-weights w_v for all $v \in V$, a root node $r \in V$, and a collection of M subsets of nodes $K = \{K_1, ..., K_M\}$ in $V \setminus \{r\}$ with $|K_i| = 2$ for all i, compute a subgraph H of G that contains at least 2-edge-disjoint paths from each group K_i to r such that the end-points of these paths in K_i are distinct. A subgraph H satisfying this property is called a 2-edge-connected group Steiner subgraph. The objective is to minimize total cost of the vertices in H.*

We refer to this problem as the node-weighted 2-edge-connected group Steiner subgraph problem (abbreviated as $NW2ECGS$). For a given instance of the problem, it is easy to determine whether there is a feasible solution by checking whether

there are at least two edge-disjoint paths in G between each group of terminals and the root vertex. This can be done in polynomial time using standard network flow techniques (see, e.g., [1]). Therefore, we can assume that the given instance of the problem admits at least one feasible solution.

Recall that the node-weighted δ-Steiner subgraph problem is formulated via a monotone property $\delta(H, K)$ that is defined in terms of a graph H and a subset K of the node set of H. Strictly speaking, the property $\chi(H, K)$ that is used to define the 2-edge-connected group Steiner subgraph problem does not fit into this framework because in the case of $\chi(H, K)$, K is a collection of subsets of nodes of H. However, the result stated as Theorem 4.1 for the node-weighted δ-Steiner subgraph problem also holds for the node-weighted 2-edge-connected group Steiner subgraph problem because the arguments used in the proofs that establish the theorem apply also if $\chi(H, K)$ is considered in place of $\delta(H, K)$.

In general graphs, the approximation ratio achieved for the node-weighted 2-edge-connected group Steiner subgraph problem is $O(\log n)$ [29]. For the edge-weighted version of the problem, there is a constant factor approximation algorithm with ratio $(2 + \rho)$ [29], where ρ is the best known approximation ratio for the edge-weighted Steiner tree problem in general graphs, currently $\rho = 1.39$. We consider $NW2ECGS$ in restricted graph classes for wireless ad-hoc networks.

The property χ is monotone, i.e., we can remove Steiner nodes of degree 1 from the solution, and we can add edges to the solution for the analysis of the algorithm when considering the induced subgraph of the solution. Removing Steiner nodes of degree 1 from the solution does not destroy the connectivity because such Steiner nodes are leaves and are not intermediate nodes. Adding edges to the solution does not destroy the connectivity of nodes either.

6.2.1 Computing 2-Edge-Connected Group Steiner Subgraphs

We adapt Algorithm 1 to compute 2-edge-connected group Steiner subgraphs with node-weights as shown in Algorithm 4.

Algorithm 4 Algorithm for $NW2ECGS$

Input: An instance of $NW2ECGS$ given by $G = (V, E)$ with nonnegative node weights w_v for $v \in V$, a root vertex r, and a collection of subsets of nodes in $V \setminus r$: $K = \{K_1, ..., K_M\}$, where $|K_i| = 2$ for all i.

Output: 2-edge-connected group Steiner subgraph H.

1 Assign weight $c(u, v)$ to every edge $(u, v) \in E$: $c(u, v) = w_u + w_v$

2 Compute a subgraph H for G satisfying $\chi(H, K)$ with respect to edge weights using the algorithm by Khandekar et al. [29].

Note that $c(u, v) = 0$ if $u, v \in \bigcup_i K_i$.

6.2.2 Analysis of Approximation Ratio

We consider only hereditary classes of graphs, i.e., if a graph is in the class, then any induced subgraph of that graph is also in the class. Moreover, we analyze the algorithm for graph classes Π that have the following property, for some constant d:

($\mathbf{P}_d^\chi$) Any graph H in the class Π with a root vertex r, and a collection of subsets of nodes in $V \setminus r$: $K = \{K_1, ..., K_M\}$ satisfying the property $\chi(H, K)$ has a spanning subgraph H' of maximum degree at most d satisfying $\chi(H', K)$.

We assume that the graph $G = (V, E)$ of the given instance of $NW2ECGS$ belongs to a class of graphs with property (P_d^χ). We prove the following:

Theorem 6.2. *For hereditary classes of graphs satisfying (P_d^χ), the above algorithm is a* $1.695d$*-approximation algorithm for the problem* $NW2ECGS$*.*

Proof. By Theorem 4.1, we have a $0.5d\rho$-approximation algorithm for the node-weighted δ-Steiner subgraph problem in hereditary classes of graphs that satisfy (P_d^δ), where δ is monotone. Although Theorem 4.1, is proved for δ-Steiner subgraphs The property $\chi(H, K)$ does not affect the proof of Theorem 4.1, The property $\chi(H, K)$ is monotone, therefore, Theorem 4.1 implies that we have a $0.5d\rho$-approximation algorithm for the node-weighted 2-edge-connected group Steiner subgraph problem in classes of graphs that satisfy (P_d^χ). The algorithm by Khandekar et al. is a 3.39-approximation algorithm for the edge-weighted version of the 2-edge-connected group Steiner subgraph problem. Thus, the approximation ratio of our algorithm is $0.5d \cdot 3.39 = 1.695d$ for the node-weighted 2-edge-connected group Steiner subgraph problem. □

6.2.3 Node-Weighted 2-Edge-Connected Group Steiner Subgraph Problem in $(\lambda + 1)$-Claw Free Graphs

Similar to how we have shown previously that there exist 2-edge-connected spanning subgraphs of maximum vertex degree d for any 2-edge-connected $(\lambda+1)$-claw-free graph, we can compute 2-edge-connected group Steiner subgraphs of bounded degree for any 2-edge-connected group Steiner subgraphs using a modified version of the algorithm given in the proof of Lemma 10.

Lemma 11. *For every* $\lambda \geq 1$*, every* 2*-edge-connected group Steiner subgraph in the class of* $(\lambda + 1)$*-claw-free graphs has a subgraph that is also a 2-edge-connected group Steiner subgraph and has maximum degree at most* $2(\lambda + 1)$*.*

Proof. Assume that G is a 2-edge-connected group Steiner subgraph in the class of $(\lambda+1)$-claw-free graphs. We construct a subgraph G' using the algorithm presented

in the proof of Lemma 10 with one modification: If, in step 6, there is no edge connecting v or a node above v to a node in the subtree below c, the algorithm does not add an edge. We first compute a DFS tree T starting with vertex r. Then edges are added to T to get a subgraph G'. We claim that in G' there are 2 edge-disjoint paths from every group K_i to r. Note that every $v \in K_i$ for any group K_i has a path to r in G' because G' contains the spanning tree T. Consider a cut C that separates r and K_i in G'. Let u-v be an edge going across this cut. Suppose u and r are located on one side of the cut and v and K_i are on the other side of the cut. If u-v is the only edge that crosses C in G', the v-side of the cut C must consist of v and all its descendants in T (and no other node). Because G is a feasible solution, there must be another edge from u or an ancestor of u to v or a descendant of v. The algorithm would have added such an edge to G' when processing u-v. Therefore, u-v cannot be the only edge across C in G'. By the same argument as in Lemma 10, it follows that G' has degree at most $2(\lambda+1)$. □

By applying Theorem 6.2 with $d = 2(\lambda + 1)$, we obtain the following corollary.

Corollary 9. *There is an approximation algorithm with approximation ratio* $1.695 \cdot 2(\lambda+1) = 3.39(\lambda+1)$ *for the node-weighted* 2*-edge-connected group Steiner subgraph problem in* $(\lambda + 1)$*-claw-free graphs.*

6.2.4 *NW2ECGS* in Special Graph Classes

The result obtained in Corollary 9 for $(\lambda+1)$-claw-free graphs can be used to derive results for other graph classes.

In unit disk graphs, the number of independent neighbors of every node is at most 5 [59], therefore, we have the following result:

Corollary 10 (UDG)**.** *There is a* $1.695 \cdot 12 = 20.34$*-approximation algorithm for the NW2ECGS problem in unit disk graphs.*

α-unit disk graphs are $(4/\alpha^2 + 4/\alpha + 2)$-claw-free due to Lemma 5, therefore, we have:

Corollary 11 (α-UDG)**.** *There is a* $1.695 \cdot 2 \cdot (4/\alpha^2 + 4/\alpha + 2) = 3.39 \cdot (4/\alpha^2 + 4/\alpha + 2)$*-approximation algorithm for the* $NW2ECGS$ *problem in* α*-unit disk graphs.*

Bi-directional disk graphs are $(18\lceil \log_2 m \rceil + 13)$-claw-free (see Section 5.1.1), hence we have the following corollary:

Corollary 12 (bi-directional disk graphs)**.** *There is a* $1.695 \cdot 2 \cdot (18\lceil \log_2 m \rceil + 13) = 3.39 \cdot (18\lceil \log_2 m \rceil + 13)$*-approximation algorithm for the* $NW2ECGS$ *problem in bi-directional disk graphs, for* $m > 1$*.*

6.3 Node-Weighted 2-Vertex-Connected Steiner Subgraph Problem

We also consider 2-vertex-connected Steiner subgraphs, i.e., subgraphs in which there are at least 2 vertex-disjoint paths between every pair of vertices.

We consider the property $\varsigma(H, K)$ which represents that H is a 2-vertex connected graph containing all vertices in K. We define the node-weighted 2-vertex-connected Steiner subgraph problem ($NW2VCS$) as follows:

Problem 5. *Given an undirected graph* $G = (V, E)$ *with nonnegative weights* w_v *for* $v \in V$*, and a subset of nodes* $K \subseteq V$ *(*terminals*), compute a subgraph* H *for* G *and* K *that satisfies the property* $\varsigma(H, K)$*. The goal of the* $NW2VCS$ *problem is to minimize the total weight of the vertices of* H*.*

Requiring only that H has 2 vertex disjoint paths between every pair of terminals does not change the above problem.

The property ς is monotone as we could add edges to H for the analysis of our algorithm, and also we could delete Steiner nodes of degree 1, in fact, a 2-vertex-connected Steiner subgraph cannot contain a vertex of degree less than 2.

Using network flow techniques [1], we can check in polynomial time whether a given instance of $NW2VCS$ admits at least one feasible solution. Therefore, we assume that the given instance of the $NW2VCS$ has at least one feasible solution.

6.3.1 Computing 2-Vertex-Connected Steiner Subgraphs

We adapt Algorithm 1 to compute 2-vertex-connected Steiner subgraphs with node-weights as shown in Algorithm 5.

Algorithm 5 Algorithm for $NW2VCS$

Input: An instance of $NW2VCS$ given by $G = (V, E)$ with nonnegative node weights w_v for $v \in V$ and a set $K \subseteq V$ of terminals.
Output: 2-vertex-connected Steiner subgraph H.

1 Assign weight $c(u, v)$ to every edge $(u, v) \in E$: $c(u, v) = w_u + w_v$
2 Compute a subgraph H for G satisfying $\varsigma(H, K)$ with respect to edge weights using the algorithm by Fleischer et al. [17].

We assume that $w_u = 0$ for all $u \in K$, thus $c(u, v) = 0$ if $u, v \in K$.

6.3.2 Analysis of Approximation Ratio

For some constant d, we analyze Algorithm 5 for graph classes that have the property ($\mathrm{P}^{\varsigma}_{d}$):

($\mathbf{P}^{\varsigma}_{d}$) Any 2-vertex-connected graph in the class contains a spanning 2-vertex-connected subgraph of maximum vertex degree d.

Let graph $G = (V, E)$ of the given instance of $NW2VCS$ belong to a class of graphs with property ($\mathrm{P}^{\varsigma}_{d}$). The result for the node-weighted δ-Steiner subgraph problem in Chapter 4 is adapted for the $NW2VCS$ problem:

Theorem 6.3. *Algorithm 5 described above is a d-approximation algorithm for the problem NW2VCS in hereditary classes of graphs satisfying property (P_d^{ς}).*

Proof. We know that property $\varsigma(H, K)$, which models 2-vertex connected Steiner subgraphs, is monotone. Furthermore, as proved in Theorem 4.1, we have a $0.5d\rho$-approximation algorithm for the node-weighted δ-Steiner subgraph problem in hereditary classes of graphs that satisfy (P_d^{δ}), where δ is monotone. Therefore, we have a $0.5d\rho$-approximation algorithm for the node-weighted 2-edge-connected Steiner subgraph problem in classes of graphs that satisfy (P_d^{ς}). The algorithm presented by Fleischer et al. [17] has approximation ratio 2, thus we have an approximation algorithm with ratio $0.5d \cdot 2 = d$ for the node-weighted 2-vertex-connected Steiner subgraph problem. □

6.3.3 Node-Weighted 2-Vertex-Connected Steiner Subgraphs in $(\lambda + 1)$-claw-free graphs

The following lemma shows that $(\lambda+1)$-claw-free graphs satisfy the property $(\mathrm{P}_d^{\varsigma})$.

Lemma 12. *For $\lambda \geq 1$, every 2-vertex-connected $(\lambda + 1)$-claw-free graph has a 2-vertex connected spanning subgraph of maximum degree at most $2(\lambda + 1)$.*

Proof. In order to analyze the maximum degree bound of every node in a 2-vertex-connected spanning subgraph, we present an algorithm that computes a DFS tree T for a connected $(\lambda + 1)$-claw-free graph $G = (V, E)$, and then adds more edges to make the graph 2-vertex-connected.

1. Compute a DFS tree T for graph $G = (V, E)$
2. $G' = T$
3. Process tree T top-down: for each node $v \in V$

4. for all children c of v in T

5. if c and the parent of v in T are in different components of G'-v then

6. add to G' an edge from a node above v to a node in the subtree below c, with deepest lower end-point.

Every node in a $(\lambda+1)$-claw-free graph can have at most λ independent neighbors. Moreover, we know that the children of every node in a DFS tree are independent. Therefore, in the DFS tree T, every node has at most λ children. When we add edges to make T 2-vertex-connected, each node gets at most $(\lambda + 1)$ new neighbors. In the following, we show that the degree of each node is at most $2(\lambda+1)$ in the end.

Consider an arbitrary node w, and assume that the algorithm adds an edge from above with lower end-point w, which is the first new edge added to w. Assume t is the upper end-point of this added edge. Let us assume that this edge is added when the parent and at least one child of a node u are in different components of $G' - u$. Suppose v is such a child of u. After addition of the edge from t to w in the subtree below v, all nodes on the path between t and w are 2-vertex-connected. Therefore, the algorithm does not add a second edge from above to w.

We claim that every node can get at most λ new neighbors that are below it. This is due to the fact that among all new neighbors added to a node w from below, none of the nodes is an ancestor of another. If such a situation arises, then we have a contradiction that an edge has been added from w to its deepest descendent. Hence, we can say that all new neighbors added to w from below are independent nodes, and the number of these nodes is at most λ.

Now, the initial number of w's neighbors in T (when no extra edge is added yet for 2-vertex-connectivity), plus the number of neighbors added to w from above

(ancestors) and below (descendants) in T, gives us the degree of w in the end. Thus, w has at most $2(\lambda + 1)$ neighbors in the 2-vertex-connected spanning subgraph G'. □

Theorem 6.3 and Lemma 12 imply the following result:

Corollary 13. *There is a* $2(\lambda + 1)$*-approximation algorithm for the* $NW2VCS$ *problem in* $(\lambda + 1)$*-claw-free graphs.*

In the following subsection we present results obtained for the node-weighted 2-vertex-connected Steiner subgraph problem in graph classes that are included in the class of $(\lambda + 1)$-claw-free graphs.

6.3.4 $NW2VCS$ in Other Graph Classes

Since unit disk graphs are 6-claw-free [59], quasi-unit disk graphs are $(4/\alpha^2 + 4/\alpha + 2)$-claw-free (due to Lemma 5), and bi-directional disk graphs are $(18\lceil\log_2 m\rceil + 13)$-claw-free (see Section 5.1.1), we have the following corollaries:

Corollary 14 (UDG)**.** *There is a* 12*-approximation algorithm for the* $NW2VCS$ *problem in unit disk graphs.*

Corollary 15 (α-UDG)**.** *There is a* $2 \cdot (4/\alpha^2 + 4/\alpha + 2)$*-approximation algorithm for the* $NW2VCS$ *problem in* α*-unit disk graphs.*

Corollary 16 (bi-directional disk graphs)**.** *There is a* $2\cdot(18\lceil\log_2 m\rceil+13)$*-approximation algorithms for the* $NW2VCS$ *problem in bi-directional disk graphs, for* $m > 1$*.*

Conclusion

Using algorithms analogous to the algorithm for the node-weighted δ-Steiner subgraph problem, we have obtained good approximation results for node-weighted

versions of the 2-edge-connected Steiner subgraph problem, the 2-edge-connected group Steiner subgraph problem, and the 2-vertex-connected Steiner subgraph problem in restricted graph classes for wireless ad-hoc networks. We have shown that there exist solutions of bounded maximum degree for our problem variants in $(\lambda+1)$-claw-free graphs and other special graph classes considered in this thesis.

Chapter 7

Node-Weighted k-Edge-Connected Steiner Subgraph Problem

In the framework of related graph problems motivated by wireless ad-hoc networks, we continue to study Steiner subgraphs in the classes of graphs that are employed to model wireless ad-hoc networks. The problem variant that we consider in this chapter is the node-weighted version of the k-edge-connected Steiner subgraph problem. A graph is said to be k-edge-connected if all vertices are connected by k edge-disjoint paths. For example in Fig 7.1, every pair of vertices can communicate via at least 2 edge-disjoint paths, so the graph is 2-edge-connected. We consider

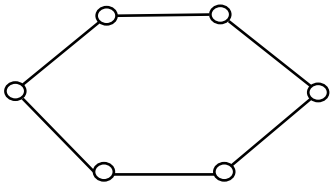


Figure 7.1: Every vertex is 2-edge-connected.

the property $\psi_k(H, K)$ which specifies that H is a graph that contains all vertices in K and has at least k edge-disjoint paths between every pair of vertices in K. The node-weighted k-edge-connected Steiner subgraph problem (abbreviated as

$NWkECS$) is defined as follows:

Problem 6. *Given an undirected graph $G = (V, E)$ with nonnegative weights w_v for $v \in V$, and a subset of nodes $K \subseteq V$, compute a subgraph H for G and K that satisfies the property $\psi_k(H, K)$. The objective is to minimize the total weight of the vertices of H.*

A subgraph H satisfying $\psi_k(H, K)$ is called k-edge-connected Steiner subgraph. The considered property $\psi_k(H, K)$ satisfies monotonicity, i.e., we can add an edge to H and also we can delete a Steiner node of degree 1 from H, and by doing so the property $\psi_k(H, K)$ is not violated. We can assume that H does not contain any Steiner nodes of degree less than 2 because such vertices could be removed from the solution while maintaining feasibility. Furthermore, when we add edges to H for the analysis of the algorithm, it does not destroy the property of k-edge-connectivity.

Before we present the algorithm and its analysis for the k-edge-connected Steiner subgraph problem, we prove that the k-edge-connected Steiner subgraph problem is NP-hard in unit disk graphs for $k \geq 2$. For $k = 1$, the problem is equivalent to the Steiner tree problem, which is already known to be NP-hard [11].

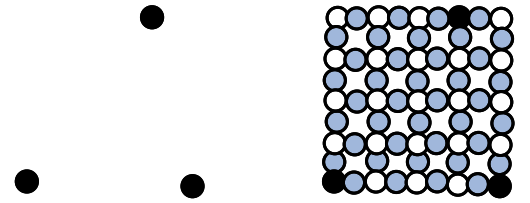
7.1 $NWkECS$ is NP-hard in UDG

Theorem 7.1. *The node-weighted k-edge-connected Steiner subgraph problem is NP-hard in unit disk graphs for $k \geq 2$.*

Proof. The rectilinear Steiner tree problem in which all terminals have integer coordinates that are polynomially bounded, is NP-hard as shown by Garey and Johnson [19]. To prove that the k-edge-connected Steiner subgraph problem is NP-hard in UDG for $k \geq 2$, we reduce the rectilinear Steiner tree problem to the

k-edge-connected Steiner subgraph problem.

Consider an instance of the rectilinear Steiner tree problem having terminals at polynomially bounded integer coordinates and denote it by I. We construct an instance of the k-edge-connected Steiner subgraph problem for a unit disk graph denoted by I' from I as follows:



(a) An instance of the rectilinear Steiner tree problem I.

(b) constructed instance of the k-edge-connected Steiner subgraph problem I' from I.

Figure 7.2: Black nodes represent terminals, white nodes are Steiner nodes of weight 1 and blue nodes are Steiner nodes of weight 0.

If there is a terminal at an integer coordinate (i, j), we put a terminal in the UDG at the same position; otherwise, we put a unit disk of weight 1 only for the integer coordinate points within the bounding rectangle of the terminals (see Figure 7.2). In between any two vertically or horizontally adjacent disks, we put k unit disks of weight 0 such that each of the k disks is a neighbor of these two vertically or horizontally adjacent disks but not connected to any other disks. We use disks of radius 0.3 to construct I'.

We claim that there is a k-edge-connected Steiner subgraph T' for I' of weight $W \leq L + 1 - |K|$ if and only if there exists a rectilinear Steiner tree T for I with length at most L, where $|K|$ is the number of terminals. See Fig. 7.3 for an illustration. Let T be a rectilinear Steiner tree for I with length L, and a *segment* be a horizontal or vertical line between two neighboring integer coordinate points.

We can assume that T consists only of such segments (i.e., all lines in T are at integer x- or y-coordinates). The total length L of T is the sum of the lengths of all segments in T. For every segment in T, add to T' the unit disks at both ends of the segment and the k unit disks in between. Since terminals have weight zero, W is the weight of T' contributed by unit disks of weight 1. Alternatively, we can say that the total number of Steiner nodes of weight 1 in T' is the total weight of T'. Since we have constructed T' from T, and we know that each Steiner node of weight 1 in T' corresponds to an integer coordinate point in T that does not have a terminal, the total number of integer coordinate points minus the number of terminals in T is the total number of Steiner nodes in T'. $L + 1$ is the total number of integer coordinate points in T. So, the total number of Steiner nodes of weight 1 in T' or the total weight W of T' satisfies $W \leq L + 1 - |K|$.

Conversely, suppose that there is a Steiner subgraph T' for I' of weight W, which is the sum of the weights of the Steiner nodes in T'. We can assume that T' contains a disk in the middle of a segment only if it also contains the disks at both end points of the segment. We construct T from T' by choosing all line segments where at least one of the k disk in the middle is chosen for T'. The length of T is equal to the total number of integer coordinate points included in T minus 1. We know that each integer coordinate point in T corresponds either to a Steiner node of weight 1 or a terminal in T', therefore, the length of T is equal to the sum of the weights of the Steiner nodes and the number of terminals in T' minus 1: $L = W + |K| - 1$.

We have shown that the problem of deciding whether I has a solution of length L can be reduced in polynomial time to the problem of deciding whether I' has a solution of weight at most $L-|K|+1$. This implies that $NWkECS$ is NP-hard. □

The choice of 0.3 as disk radius in the reduction above can be explained as

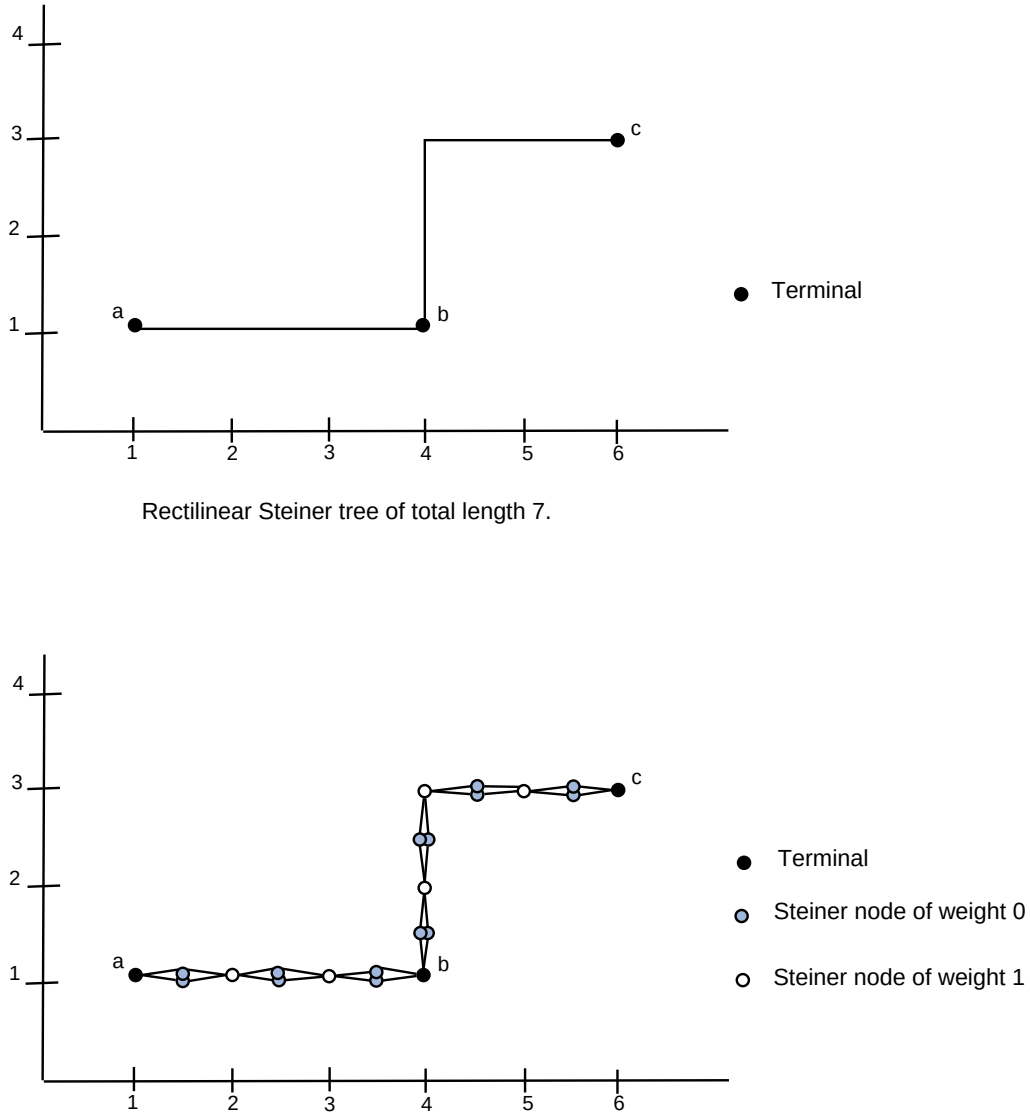
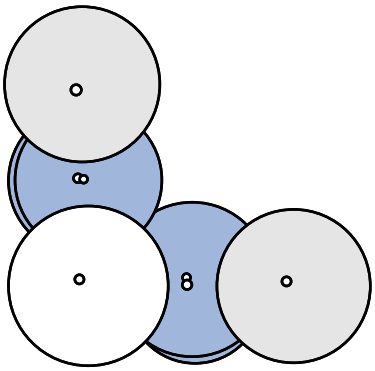
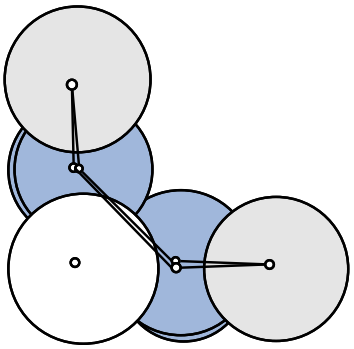


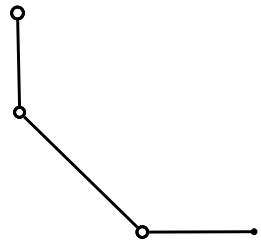
Figure 7.3: Example showing a rectilinear Steiner tree and its corresponding k-edge-connected Steiner subgraph



(a) An instance of the k-edge-connected Steiner subgraph problem G', for $k = 2$

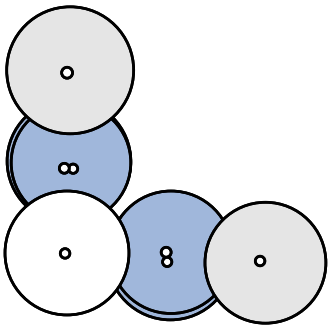


(b) H', A solution for G'

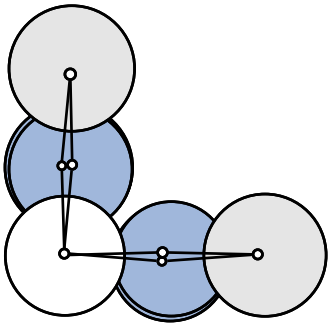


(c) corresponding Steiner tree for the above solution

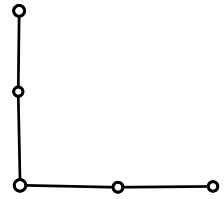
Figure 7.4: Example showing disks of radius larger than $\frac{1}{4}\sqrt{2} \approx 0.35$. Grey disks are terminals, white disks are Steiner nodes of weight 1 located at integer coordinate points, and blue disks are Steiner nodes of weight zero located between the integer coordinate points.



(a) An instance of the k-edge-connected Steiner subgraph problem G', for $k = 2$



(b) H', a solution of G'



(c) corresponding Rectilinear Steiner tree for the above solution

Figure 7.5: Example showing disks of radius smaller than $\frac{1}{4}\sqrt{2} \approx 0.35$. Grey disks are terminals, white disks are Steiner nodes of weight 1 located at integer coordinate points, and blue disks are Steiner nodes of weight zero located between the integer coordinate points.

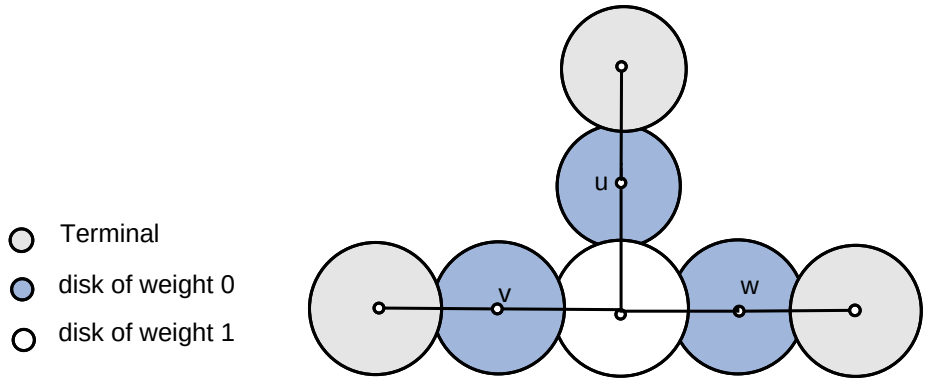


Figure 7.6: Radius 'r' of each disk is 0.3. Distance from u to v and also from u to w should be larger than $2r$ to avoid edges uv and uw.

follows: If we choose the radius of each disk larger than $\frac{1}{4}\sqrt{2} \approx 0.35$, then there is a situation that k disks at one coordinate point are adjacent to k disks located at another point. Consider the instance of the k-edge-connected Steiner subgraph problem G' in Fig. 7.4. Assume H' is a solution for G'. We aim to construct a rectilinear Steiner tree from H' in the same way we constructed T' in the above proof. Grey disks are terminals, white disks represent Steiner nodes of weight 1 located at integer coordinate points, and blue disks are Steiner nodes of weight zero located between the integer coordinate points. We can construct a tree from H' but that is not a rectilinear Steiner tree because it contains a diagonal edge. If we choose the radius of each disk equal to r, then the distance from u to w should be greater than $2r$ in order to avoid edges like u-w (see Fig. 7.6). When $r = 0.5$, the distance from u to w is $\sqrt{(0.5)^2 + (0.5)^2} = \frac{1}{2}\sqrt{2} \approx 0.707$ which is less than $2r$. Therefore, we must have that $\sqrt{(0.5)^2 + (0.5)^2} > 2r$ which implies that r should be less than $\frac{1}{4}\sqrt{2} \approx 0.35$. Also, we need radius of each disk at least 0.25.

When we take disks of radius smaller than $\frac{1}{4}\sqrt{2} \approx 0.35$ (see Fig. 7.5), we get a rectilinear Steiner tree for a solution of the given instance of the k-edge-connected Steiner subgraph problem.

7.2 Algorithm for the Node-Weighted k-Edge-Connected Steiner Subgraph Problem

Algorithm 1 can be adapted to the node-weighted k-edge-connected Steiner subgraph problem as shown in Algorithm 6.

Algorithm 6 Algorithm for *NWkECS*

Input: An instance of *NWkECS* given by $G = (V, E)$ with nonnegative node weights w_v for $v \in V$ and a set $K \subseteq V$ of terminals.
Output: k-edge-connected Steiner subgraph H.

1 Assign weight $c(u, v)$ to every edge $(u, v) \in E$: $c(u, v) = w_u + w_v$
2 Compute a subgraph H for G satisfying $\psi_k(H, K)$ with respect to edge weights using the algorithm by Jain [26].

We assume that every terminal has weight 0, therefore, $c(u, v) = 0$ if $u, v \in K$. The edge-weighted version of the k-edge-connected Steiner subgraph problem is a special case of the generalized Steiner network problem: $r_{i,j}$ equals k if $i, j \in K$ and 0 otherwise (see Chapter 3 for a definition of the generalized Steiner network problem). Therefore, we use the best known approximation algorithm for the generalized Steiner network problem, i.e., the algorithm by Jain [26]. The approximation ratio of that algorithm is 2.

7.2.1 Analysis of the Approximation Ratio

For analysis of Algorithm 6, we consider only hereditary classes of graphs, i.e., if a graph is in the class, then any induced subgraph of that graph is also in the class. Furthermore, we consider graph classes Π that have the following property, for some constant d:

($\mathbf{P}_d^{\psi_k}$) Any graph H in the class Π, and a subset of vertices $K \subseteq V(H)$ satisfying

the property $\psi_k(H, K)$ has a subgraph H' of maximum degree at most d satisfying $\psi_k(H', K)$.

Assume that the graph $G = (V, E)$ of the given instance of $NWkECS$ belongs to a class of graphs with property ($\mathrm{P}_d^{\psi_k}$). Then, we have:

Theorem 7.2. *For hereditary classes of graphs satisfying ($P_d^{\psi_k}$), the algorithm described above is a d-approximation algorithm for $NWkECS$.*

Proof. By Theorem 4.1, we have a $0.5d\rho$-approximation algorithm for the node-weighted δ-Steiner subgraph problem in hereditary classes of graphs that satisfy (P_d^{δ}), where δ is monotone. We know that the property $\psi_k(H, K)$ is monotone, therefore, Theorem 4.1 implies that we have a $0.5d\rho$-approximation algorithm for the node-weighted k-edge-connected Steiner subgraph problem in classes of graphs satisfying ($P_d^{\psi_k}$). Since Jain's algorithm has approximation ratio 2, we have an approximation algorithm with ratio $0.5d \cdot 2 = d$ for the node-weighted k-edge-connected Steiner subgraph problem. □

We show that there exists a k-edge-connected subgraph of maximum degree bounded by d in the class of $(\lambda + 1)$-claw-free graphs for any k-edge-connected Steiner subgraph. We first present an algorithm to compute a k-edge-connected subgraph in $(\lambda+1)$-claw-free graphs and then show the maximum degree bound d.

7.2.2 Computing k-Edge-Connected Subgraphs in $(\lambda + 1)$-claw-free Graphs

The following algorithm computes a subgraph by repeatedly finding forests F_i in $G - \bigcup_{j=1}^{i-1} F_j$ for $i = 1, 2, ..., k$. So the output of the algorithm is a k-edge-connected subgraph formed by the the union of forests: $F_1 \cup F_2 \cup ... \cup F_k$.

1 i :=1, $T := \phi$;

2 while $i \leq k$ do

begin

3 Compute spanning forest F_i in $G - T$ using DFS

4 $T := F_i \cup T$

5 $i := i + 1$

end

6 output T

Due to the following Lemma [3], we know that the spanning subgraph we compute has k edge-disjoint paths between every pair of terminals.

Lemma 13. (Nagamochi and Ibaraki [42]) *For a graph $G = (V, E)$, let $F_i = (V, E_i)$ be a maximal spanning forest in $G - E_1 \cup E_2 \cup ... \cup E_i$, for i=1,2,..,$|E|$, where possibly that $E_i = E_{i+1} = ... = E_{|E|} = \emptyset$ for some i. Then each spanning subgraph $G_i = (V, E_1 \cup E_2 \cup .....\cup E_i)$ satisfies: $\zeta(x, y; G_i) \geq min\{\zeta(x, y; G), i\}$ for all $x, y \in V$, where $\zeta(x, y; H)$ denotes the edge-connectivity between x and y in graph H.*

Thus, a pair of vertices x and y are connected by k-edge-disjoint paths in the subgraph $G_k = (V, E')$ if they are k-edge-connected in G, where $E' = E_1 \cup E_2 \cup ... \cup E_k$.

It remains to show that the subgraph computed above has maximum degree at most d. We need to compare the number of independent neighbors of an arbitrary node in $G - (F_1 \cup F_2.... \cup F_i)$ to that in $G - (F_1 \cup .... \cup F_{i-1})$ to find the maximum degree bound d.

Lemma 14. *The number of independent neighbors of a node in $G - (F_1 \cup F_2 ... \cup F_i)$ is at most twice the number of its independent neighbors in $G - (F_1 \cup .... \cup F_{i-1})$ in $(\lambda + 1)$-claw-free graphs.*

[3] Thanks to Frank Kammer for pointing us to the result of Nagamochi and Ibaraki [42].

Proof. Let the number of independent neighbors of a node u be I in $G-(F_1 \cup F_2... \cup F_i)$. Consider u and these I neighbors in the previous iteration, i.e., $G-(F_1 \cup .... \cup F_{i-1})$. Because it is a property of forests that they are 2-colorable, we can 2-color the I neighbors of u with respect to forest F_i. Thus, at least half of I belong to one color. As vertices of the same color do not have edges between them, u has at least $I/2$ independent neighbors in $G-(F_1 \cup .... \cup F_{i-1})$. □

Lemma 15. *The degree bound of the k-edge-connected Steiner subgraph is at most* $(2^k-1)\lambda + k$ *in* $(\lambda+1)$*-claw-free graphs.*

Proof. When we construct a spanning forest by depth first search in a $(\lambda+1)$-claw-free graph $G=(V,E)$, the degree bound of every node is at most λ+1 (see proof of Lemma 6 in Section 5.1.1), where λ is the maximum number of independent neighbors in graph G. Furthermore, it follows from the previous lemma that the number of independent neighbors of an arbitrary node in $G-(F_1 \cup F_2... \cup F_{i-1})$ is at most $2^{i-1}\lambda$.

Therefore, the degree bound of the spanning forest F_i denoted by $deg(F_i)$ is given by:

$$deg(F_i) \leq 2^{i-1}\lambda + 1$$

We already know that our k-edge-connected Steiner subgraph is the union of spanning forests: $F_1 \cup ... \cup F_k$, thus the degree bound of every vertex in the k-edge-connected Steiner subgraph is as follows:

$$deg(F_1 \cup ... \cup F_k) \leq \sum_{i=1}^{k}(2^{i-1}\lambda + 1) = (2^k-1)\lambda + k$$

□

By applying Theorem 7.2 with $d \leq (2^k-1)\lambda + k$ we have the following result:

Corollary 17. *There exists an approximation algorithm with approximation ratio* $(2^k - 1)\lambda + k$ *for the* $NWkECS$ *problem in* $(\lambda + 1)$*-claw-free graphs.*

7.2.3 $NWkECS$ in Other Graph Classes

In this section, we derive results from Corollary 17 for special classes of graphs that are included in the class of $(\lambda + 1)$-claw-free graphs.

First we obtain a result for unit disk graphs, which are 6 claw-free [59]. Therefore, we have the following corollary:

Corollary 18. *There is an approximation algorithm with approximation ratio* $(2^k - 1)5 + k$ *for* $NWkECS$ *in unit disk graphs.*

For α-unit disk graphs, the maximum number of independent neighbors of a node is at most $(4/\alpha^2 + 4/\alpha + 1)$ (see Lemma 5). Thus, we have the following result for α-unit disk graphs:

Corollary 19. *There is an approximation algorithm with approximation ratio* $(2^k - 1)(4/\alpha^2 + 4/\alpha + 1) + k$ *for* $NWkECS$ *in* α*-unit disk graphs.*

Now consider bi-directional disk graphs. Every vertex has at most K independent neighbors, where $K = 6(3\lceil \log_2 m \rceil + 2)$ [53] for $m > 1$ (m is the ratio between the maximum and minimum transmission range in a bi-directional disk graph). Thus, we have the following result for $NWkECS$ in bi-directional disk graphs:

Corollary 20. *There is an approximation algorithm with approximation ratio* $(2^k - 1)(18\lceil \log_2 m \rceil + 12) + k$ *for* $NWkECS$ *in bi-directional disk graphs, where* $m > 1$.

For $\lambda = p(1)$, bounded independence graphs are $(\lambda + 1)$-claw-free (see Section 2.2). Thus, Corollary 17 implies the following result:

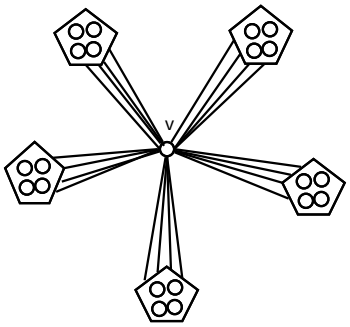


Figure 7.7: Lower bound on the degree of a k-edge-connected unit disk graph

Corollary 21. *There is an approximation algorithm with approximation ratio* $(2^k - 1)\lambda + k$ *for the NWkECS problem in bounded independence graphs.*

7.3 Lower Bound on the Maximum Degree in a k-Edge-Connected Steiner Subgraph in UDG

From the previous section, we know that the maximum degree bound d in a k-edge-connected subgraph for unit disk graphs is at most $(2^k - 1)5 + k$. Now we are interested in finding a lower bound on the maximum degree in a k-edge-connected subgraph for the class of unit disk graphs.

Lemma 16. *There exists a unit disk graph and a set of terminals such that any* k*-edge-connected Steiner subgraph has maximum degree at least* $5k$*.*

Proof. Consider a k-edge-connected unit disk graph G that is constructed as follows. There is an arbitrary vertex v in G. Terminals constituting five cliques are located around v in a 360° sector, and the size of each clique is $k + 1$ (for example see Fig. 7.7). In this construction, the degree of v is $5k$. All vertices of G are terminals.

Suppose that any two vertices, each belonging to a distinct clique, have more than 60° angle between them at v. Due to this assumption it is impossible that a pair of vertices of separate cliques have an edge between them in G. If we remove

even a single edge from v to a vertex in any of the cliques, then the property of k-edge-connectivity is lost. Hence, the degree of v which is equal to $5k$, cannot be decreased. Therefore, the lower bound on the degree of any k-edge-connected Steiner subgraph of this unit disk graph is $5k$. □

Conclusion

In this chapter, we showed that $NWkECS$ is NP-hard in unit disk graphs for $k \geq 1$. We presented an algorithm with approximation ratio $(2^k - 1)\lambda + k$ for $NWkECS$ in $(\lambda + 1)$-claw-free graphs, and then derived results for several restricted classes of graphs. Moreover, we presented a lower bound on the degree of a k-edge-connected Steiner subgraph which is only linear in k. Since the upper bound is exponential in k, there is a large gap between the lower and upper bound on the maximum degree in k-edge-connected Steiner subgraphs in unit disk graphs. In the future, it would be interesting to find better algorithms for constructing k-edge-connected subgraphs of small degree in unit disk graphs so that the gap between the lower and upper bound is reduced.

Chapter 8

Conclusion and Future Work

Our motivation for the study of node-weighted Steiner subgraph problems is multicast communication in wireless ad-hoc networks. All problem variants studied in this thesis are NP-hard, and thus, it is unlikely that polynomial-time optimal algorithms exist for these problems. We used approximation algorithms to find good solutions in polynomial time for these problems. We considered node-weighted graphs in order to capture the situation in which nodes have different capabilities or willingness to participate in a multicast routing due to certain constraints like energy, memory, life-time etc. Therefore, different cost values are assigned to nodes to model the above situation. We assumed that all terminals have weight zero since they are included in every solution. Our aim was to minimize the weight of the Steiner nodes because it was desirable to minimize the cost of the nodes that forward multicast packets but are not the intended transmitters or receivers in a multicast transmission themselves.

In the literature, an idealized graph model for wireless ad-hoc networks is the unit disk graph model. There are also more general graph models for these networks known as quasi-unit disk graphs, bounded independence graphs, bi-directional disk graphs. We considered all these graph classes in our work to model wireless ad-hoc

networks for the study of our optimization problems. In addition, we considered the class of $(\lambda + 1)$-claw-free graphs which includes the aforementioned graph classes, and therefore, is more general than all of them.

We addressed a generalized version of the node-weighted Steiner subgraph problems referred to as the node-weighted δ-Steiner subgraph problem. The property δ represents the edge-connectivity or node-connectivity requirement in the solution graph. We have shown that the simple approach of defining suitable edge weights for a given instance of the node-weighted δ-Steiner subgraph problem and then applying an approximation algorithm for the edge-weighted version of the δ-Steiner subgraph problem yields a good approximation ratio for the node-weighted δ-Steiner subgraph problem in graph classes that admit δ-Steiner subgraphs of small degree. The approximation ratio achieved for this problem is $0.5\rho d$, where ρ is the best known approximation ratio for the edge-weighted δ-Steiner subgraph problem, and d is the bound on the maximum degree of δ-Steiner subgraphs in the considered class of graphs. We adapt the algorithm for the node-weighted δ-Steiner subgraph problem for problem variants with the different connectivity requirements δ mentioned below.

For computing good multicast trees in wireless ad-hoc networks, we have considered the node-weighted Steiner tree problem in the previously mentioned graph models. We presented a $0.695 \cdot (\lambda + 1)$-approximation algorithm for the class of $(\lambda+1)$-claw-free graphs. For the computation of fault-tolerant routing structures in wireless networks, we have considered the node-weighted 2-edge-connected Steiner subgraph problem and presented an approximation algorithm with ratio $2(\lambda + 1)$ in $(\lambda + 1)$-claw-free graphs. The study of higher edge-connectivity requirement was extended to k-edge-connected Steiner subgraphs with node weights. A d-approximation algorithm was presented for the node-weighted k-edge-connected Steiner subgraph problem in $(\lambda+1)$-claw-free graphs, where $d \leq (2^k - 1)\lambda + k$ is a

bound on the maximum degree of a k-edge-connected solution subgraph in $(\lambda+1)$-claw-free graphs. We also considered the node-weighted 2-vertex-connected Steiner subgraph problem and 2-edge-connected group Steiner subgraph problem in special classes of graphs. For the node-weighted 2-vertex-connected Steiner subgraph problem, we presented a $2(\lambda+1)$-approximation algorithm in $(\lambda+1)$-claw-free graphs. For the 2-edge-connected group Steiner subgraph problem, we achieved approximation ratio $3.39(\lambda+1)$ in $(\lambda+1)$-claw-free graphs. The key point in the analysis of all of the above algorithms was to show that for the considered classes of graphs, there exist solutions of small node degree.

Our algorithms are centralized, but distributed algorithms for the edge-weighted versions of the considered problems can be used to obtain distributed implementations of our algorithms. If the approximation ratio of the distributed algorithm for the edge-weighted δ-Steiner subgraph problem is ρ, our algorithm yields approximation ratio $0.5d\rho$ for graphs having solution subgraphs with maximum degree bounded by d.

In general, for graph problems that have better approximation results for their edge-weighted version as compared to their node-weighted version in arbitrary graphs, one might use a similar approach as we have used to get better approximation results for restricted graph classes. In our approach, the main ingredient that is necessary is the existence of optimal solution subgraphs of small node degree. If this property can be shown to hold for another problem in a restricted class of graphs, then our approach may lead to a better approximation ratio for the node-weighted version of that problem in that restricted class of graphs.

Future work

One interesting question for future work is whether the approximation ratio for the node-weighted Steiner tree problem in graph models of wireless ad-hoc networks can

be improved further and whether even a polynomial-time approximation scheme exists for the problem. Similarly, one might try to improve the approximation ratio for the node-weighted 2-edge-connected, k-edge-connected, and 2-vertex-connected Steiner subgraph problems. The approximation ratios given in this thesis could be improved if we could further minimize the bounded maximum degree d of solution subgraphs in the considered graph classes.

It would be interesting to study other variants of fault tolerance requirements in restricted classes of graphs, e.g., k-vertex-connected Steiner subgraphs such that there are at least k internally vertex disjoint paths between every pair of terminals. Another interesting problem variant is to compute k-vertex-connected Steiner subgraphs in which every pair of vertices has k internally vertex disjoint paths.

One direction for future work is to study inapproximability for the connectivity problems addressed in this thesis. We have shown a lower bound on the maximum degree bound in k-edge-connected unit disk graphs. In the future, it would be interesting to find such lower bounds in other classes of graphs we considered.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

[2] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1999.

[3] S. Banerjee, A. Misra, J. Yeo, and A. Agrawala. Energy-efficient broadcast and multicast trees for reliable wireless communication. In *Wireless Communications and Networking, WCNC 2003*, volume 1, pages 660–667, March 2003.

[4] L. Barrière, P. Fraigniaud, L. Narayanan, and J. Opatrny. Robust position-based routing in wireless ad hoc networks with irregular transmission ranges. *Wireless Communications and Mobile Computing*, 3(2):141–153, 2003.

[5] M. Bern and P. Plassmann. The steiner problem with edge lengths 1 and 2,. *Inf. Process. Lett.*, 32:171–176, September 1989.

[6] J. L. Bredin, E. D. Demaine, M. Taghi Hajiaghayi, and D. Rus. Deploying sensor networks with guaranteed fault tolerance. *IEEE/ACM Transactions on Networking*, 18:216–228, February 2010.

[7] V. Bryant, editor. *Metric Spaces*. Cambridge University Press, New York, NY, USA, 1985.

[8] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved LP-based approximation for Steiner tree. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC 2010, pages 583–592, New York, NY, USA, 2010. ACM.

[9] M. Chudnovsky and P. Seymour. The structure of claw-free graphs. In *Proceedings of the British Combinatorial Conference*, pages 153–172. Cambridge University Press, 2005.

[10] J. Chuzhoy and S. Khanna. An $O(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 437–441. IEEE Computer Society, 2009.

[11] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

[12] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[13] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, Third edition, 2005.

[14] D. Dreyer and M. Overton. Two heuristics for the Euclidean Steiner tree problem. *Journal of Global Optimization*, 13:95–106, 1998.

[15] T. Erlebach and A. Shahnaz. Approximating node-weighted multicast trees wireless ad hoc networks. In *Proceedings of the 5th International Conference*

on Wireless Communication and Mobile Computing, IWCMC 2009, pages 639–643, 2009.

[16] J. Fakcharoenphol and B. Laekhanukit. An $O(\log^2 k)$-approximation algorithm for the k-vertex connected spanning subgraph problem. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC 2008, pages 153–158, New York, NY, USA, 2008. ACM.

[17] L. Fleischer, K. Jain, and D. P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *Journal of Computer and System Sciences*, 72:838–867, August 2006.

[18] A. Frieze, J. Kleinberg, R. Ravi, and W. Debany. Line-of-sight networks. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2007, pages 968–977, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[19] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32:826–834, 1977.

[20] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[21] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms*, 37:66–84, October 2000.

[22] A. M. Gibbons. *Algorithmic graph theory*. Cambridge Univ Pr, 1985.

[23] M. X. Goemans and D. J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60:145–166, June 1993.

[24] M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. *Handbook in Operations Research and Management Science*, pages 617–671, 1995.

[25] S. Guhaa and S. Khuller. Improved methods for approximating node weighted Steiner trees and connected dominating sets. *Information and Computation*, 150(1):57–74, 1999.

[26] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21:39–60, 2001.

[27] L. Kamma and Z. Nutov. Approximating survivable networks with minimum number of Steiner points. In Klaus Jansen and Roberto Solis-Oba, editors, *Approximation and Online Algorithms*, volume 6534 of *Lecture Notes in Computer Science*, pages 154–165. Springer Berlin/Heidelberg, 2011.

[28] M. Karpinski and A. Zelikovsky. New approximation algorithms for the Steiner tree problem. *Journal of Combinatorial Optimization*, 1:47–66, 1997.

[29] R. Khandekar, G. Kortsarz, and Z. Nutov. Approximating fault-tolerant group-Steiner problems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 263–274, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[30] P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *Journal of Algorithms*, 19(1):104–115, 1995.

[31] G. Kortsarz, R. Krauthgamer, and J. Lee. Hardness of approximation for vertex-connectivity network-design problems. In Klaus Jansen, Stefano Leonardi, and Vijay Vazirani, editors, *Approximation Algorithms for Combinatorial Optimization*, volume 2462 of *Lecture Notes in Computer Science*, pages 185–199. Springer Berlin / Heidelberg, 2002.

[32] G. Kortsarz and Z. Nutov. Approximating k-node connected subgraphs via critical graphs. *SIAM Journal on Computing*, 35:247–257, July 2005.

[33] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In Pierre Fraigniaud, editor, *Distributed Computing*, volume 3724 of *Lecture Notes in Computer Science*, pages 273–287. Springer Berlin / Heidelberg, 2005.

[34] F. Kuhn, T. Nieberg, T. Moscibroda, and R. Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, DIALM-POMC 2005, pages 97–103, New York, NY, USA, 2005. ACM.

[35] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-hoc networks beyond unit disk graphs. *Wireless Networks*, 14(5):715–729, 2008.

[36] G. Markowsky L. Kou and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15(2):141–145, 1981.

[37] W. Liang. Approximate minimum-energy multicasting in wireless ad-hoc networks. *IEEE Transactions on Mobile Computing*, 5(4):377–387, 2006.

[38] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995.

[39] M. Min, H. Du, X. Jia, C. Huang, S. Huang, and W. Wu. Improving construction for connected dominating set with Steiner tree in wireless sensor networks. *Journal of Global Optimization*, 35:111–119, 2006.

[40] C. Siva Ram Murthy and B. S. Manoj. *Ad-Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[41] J. Mycielski. Sur le coloriage des graphs. *Colloquium Mathematicum*, 3(2):161–162, 1955.

[42] H. Nagamochi and T. Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(1):583–596, 1992.

[43] Z. Nutov. Approximating Steiner networks with node weights. In Eduardo Laber, Claudson Bornstein, Loana Nogueira, and Luerbio Faria, editors, *LATIN 2008: Theoretical Informatics*, volume 4957 of *Lecture Notes in Computer Science*, pages 411–422. Springer Berlin/Heidelberg, 2008.

[44] Z. Nutov. An almost $O(\log k)$-approximation for k-connected subgraphs. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2009, pages 912–921, 2009.

[45] D. Panigrahi. Survivable network design problems in wireless networks. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1014–1027, 2011.

[46] H. J. Prömel and A. Steger. *The Steiner tree problem: a tour through graphs, algorithms, and complexity*. Advanced Lectures in Mathematics. Friedrich Vieweg Sohn Verlagsgesellschaft mbH, 2002.

[47] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000*, pages 770–779, 2000.

[48] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM 1999*, pages 207–218, 1999.

[49] S. Schmid and R. Wattenhofer. Algorithmic models for sensor networks. In *Proceedings of the 20th International Conference on Parallel and Distributed Processing*, IPDPS 2006, page 177, Washington, DC, USA, 2006. IEEE Computer Society.

[50] A. Shahnaz and T. Erlebach. Approximating fault-tolerant Steiner subgraphs in heterogeneous wireless networks. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference, IWCMC 2010*, pages 529–533, 2010.

[51] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.

[52] C. Tang, C. S. Raghavendra, and V. Prasanna. Energy efficient adaptation of multicast protocols in power controlled wireless ad hoc networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks, I-SPAN 2002*, pages 80–85, 2002.

[53] M. T. Thai and D.-Z. Du. Connected dominating sets in disk graphs with bidirectional links. *IEEE Communications Letters*, 10(3):138–140, March 2006.

[54] S. Toumpis and D. Toumpakaris. Wireless ad hoc networks and related topologies: applications and research challenges. *Elektrotechnik und Informationstechnik*, 123:232–241, 2006.

[55] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[56] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 585–594, 2000.

[57] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 2011.

[58] C. W. Wu and Y. C. Tay. Amris: a multicast protocol for ad hoc wireless networks. In *Proceedings of the Military Communications Conference. MILCOM 1999. IEEE,*, volume 1, pages 25–29, 1999.

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

[60] C. Xu, Y. Xu, and J. Wu. On the minimization of the number of forwarding nodes for multicast in wireless ad hoc networks. In *Proceedings of ICCNMC 2005*, LNCS 3619, pages 286–294. Springer, 2005.

[61] A. Zelikovsky. An 11/6 approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993.

[62] F. Zou, X. Li, D. Kim, and W. Wu. Two constant approximation algorithms for node-weighted Steiner tree in unit disk graphs. In *Proceedings of the 2nd International Conference on Combinatorial Optimization and Applications, COCOA 2008*, LNCS 5165, pages 278–285. Springer, 2008.