Exact and Factor Two Algorithms for Broadcast Time

Narek Hovhannisyan

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Computer Science) at

Concordia University

Montréal, Québec, Canada

March 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Narek Hovhannisyan**

Entitled: **Exact and Factor Two Algorithms for Broadcast Time**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Pragasen Pillay

_____ External Examiner
Dr. Pablo Romero

_____ Examiner
Dr. Denis Pankratov

_____ Examiner
Dr. Dhrubajyoti Goswami

_____ Examiner
Dr. M. Reza Soleymani

_____ Supervisor
Dr. Hovhannes Harutyunyan

Approved by _____
Dr. Joey Paquet, Chair
Department of Computer Science and Software Engineering

April 26, 2024

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Exact and Factor Two Algorithms for Broadcast Time

Narek Hovhannisyan, Ph.D.

Concordia University, 2024

Broadcasting is a fundamental information dissemination primitive in interconnection networks, where a message is passed from one node to all other nodes in the network. Following the increasing interest in interconnection networks, extensive research was dedicated to broadcasting. Two main research goals of this area are finding inexpensive network structures that maintain efficient broadcasting and finding the broadcast time of a given network topology. In the scope of this study, we will mainly focus on determining the broadcast time of a given network. The broadcast time problem on an arbitrary network is known to be NP-hard. We consider this problem on different network topologies and settings.

We begin by studying the broadcast time problem on split graphs. First, we introduce a tight polynomial-time constant approximation algorithm for broadcasting on split graphs. Then, we study some important characteristics of an optimal broadcast scheme on split graphs and design a strategy for generating optimal broadcast schemes. We apply our findings to devise an efficient broadcasting heuristic on split graphs and on natural generalization of split graphs, called (k, l)-graphs.

Next, we study broadcasting on graphs that comprise some recursive structures. We introduce an exact polynomial-time algorithm on closed chains of rings. A closed chain of rings is a sequence of cycles, where every two consecutive cycles, and the first and the last cycles, share a common vertex. Additionally, we initiate a novel direction to designing broadcasting algorithms on recursively defined graphs. We provide a theoretical foundation for future broadcasting research, as well as discuss several practical applications of the approach we introduce.

Last, we study the problem on k-path graphs, one of the simpler graph families with intersecting cycles. To better understand the challenges of broadcasting on arbitrary graphs, families with

intersecting cycles are crucial to study. We improve the current best approximation ratio for broadcasting on k-path graphs by a multiplicative factor of two. Further, we propose a new optimization problem called 3-LIST-SUB, helping us to design an optimal broadcasting algorithm on restricted k-path graphs which was unsolved to date.

Acknowledgments

First and foremost, I extend my sincere appreciation to my supervisor, Professor Hovhannes Harutyunyan. Your mentorship, expertise, and dedication to academic excellence have been instrumental in shaping this research. I am truly fortunate to have had the opportunity to work under your guidance.

I would also like to acknowledge the profound impact of every teacher, professor, and lecturer who has crossed my path since first grade to this day. Your collective influence has left an indelible mark on my academic pursuits.

To my parents, Armen and Marine, I owe a debt of gratitude that words cannot adequately deliver. Your constant support and belief in my abilities have been the foundation upon which this thesis stands. Your love and encouragement have been my driving force, and I am deeply thankful for the values and resilience you instilled in me.

To my wonderful family and friends, I want to express my deep gratitude for being the architects of countless joyful memories. Thank you for being the bright and happy moments in my memories, making a beautiful picture of joy that I will always treasure.

Last but certainly not least, I extend my heartfelt thanks to my wife, Milena. Your unwavering support, understanding, and patience have been my fuel throughout this process. I can never be appreciative enough of your encouragement, love, and sacrifices. This achievement is as much yours as it is mine, and I am grateful for your enduring presence on this journey.

In acknowledging the contributions of these individuals, I am reminded of the proverb, “It takes a village”. To everyone in my incredible village of supporters; thank you all for being an integral part of this academic milestone.

Contents

List of Figures

List of Tables

Notation Table

$G(V, E)$	An undirected loop-free graph
$V(G)$	Set of vertices of graph G
$E(G)$	Set of edges of graph G
n	Number of vertices in graph G
m	Number of edges in graph G
$u \in V$	A vertex in graph G
$b(u, G)$	Broadcast time of vertex u in graph G
$b(G)$	Broadcast time of graph G
$b_S(u, G)$	Broadcast time of vertex u in graph G following scheme S
$D(G)$	Diameter of graph G
$deg(i)$	Degree of vertex i
$deg_H(i)$	Degree of vertex i in graph (subgraph) H
$\delta(G)$	Minimum degree in graph G
$\Delta(G)$	Maximum degree in graph G

All logarithms in this work are base two ($\log x = \log_2 x$), unless otherwise specified.

Chapter 1

Introduction

As the complexity of modern computational problems grows, single-processor systems, even with state-of-the-art technologies, start requiring a longer time to solve some of the arising large problems serially. One of the most common solutions to this issue was the development of multicomputer and multiprocessor systems, which can solve the problem in parallel by breaking it into smaller tasks. Parallel and distributed computing almost always contribute to achieving better completion times for modern problems, potentially securing noticeable cost savings. However, to leverage the advancements in the physical layer, it is required to devise parallel and distributed algorithms that optimally distribute data and responsibility among all processing elements to achieve better processing and communication times. After the task decomposition of the problem is determined, all processing units can execute their part of the algorithm simultaneously. However, in almost every case (other than some embarrassingly parallel problems[1]), in the course of the algorithm execution, independent processing elements need to exchange data. This can be achieved either through a shared memory or an interconnection network. Shared memory multicomputers (Shared Memory Model) have a limitation on the number of processors that can be connected together. Hence, it loses practicality as the number of processors grows. A more feasible approach to designing multicomputers is to assign each processor a separate memory and facilitate data exchange

[1] An embarrassingly parallel problem (also called embarrassingly parallelizable or perfectly parallel) is one where little or no effort is needed to separate the problem into several parallel tasks.

between the processors through message passing. In the Message Passing Model, processors interact by sending immutable messages to one another over communication channels. Interconnection networks connect nodes with some topology and accommodate communication among the nodes via communication links (channels). Each node comprises either a single or multiple processing elements.

Over the last few decades, extensive research has been dedicated to studying the characteristics of interconnection networks to determine the best communication structures for parallel and distributed computing.

There are four main collective operations for information dissemination in interconnection networks that are studied in the literature.

- One-to-one **routing**: A node sends a message to another node in the network.
- One-to-many **multicasting**: A node sends a message to some of the nodes in the network.
- One-to-all **broadcasting**: A node sends a message to the rest of the nodes in the network.
- All-to-all **gossiping**: All nodes send messages to all other nodes in the network.

Broadcasting is one of the most important information dissemination processes in an interconnected network that has been studied over the past decades. The broadcasting process starts from a single network node (referred to as the *originator*) and ends when all nodes have the information. Across the last four decades, a large number of research works have been published concerning broadcasting in networks under different models. These models differ in the number of originators, the number of receivers at each time unit, the distance of each call, the number of destinations, and other characteristics of the network. In the context of this study, we are going to focus on the classical model of broadcasting, which will be discussed in more detail in Chapter 2.

The rest of this thesis is organized as follows:

- In the next chapter, we present a literature review of some of the important results on broadcasting in general and the different network classes that have been considered.
- In Chapter 3, we study the broadcasting problem on split graphs, which are the family of graphs where the vertex set can be decomposed into a clique and an independent set. We

also extend our results on split graphs to (k, l)-graphs, which are the graphs comprising k independent sets and l cliques.

- In Chapter 4, we delve into the broadcasting problem on graphs that follow some repetitive topology patterns. We also propose a new approach for designing recursive broadcasting algorithms.

- In Chapter 5, we focus on studying broadcasting on k-path graphs. A k-path graph consists of two vertices that are connected via k internally vertex-disjoint paths. Additionally, we pose a new problem and discuss its relation to the broadcast time problem on k-path graphs.

Chapter 2

Literature Review and Preliminaries

Over the last decades, various broadcasting models have emerged in the literature. These models differ depending on the constraints placed on the originator, receiver, and message sets, as well as the network topology, the regulations of message transmissions, and/or the information about the network known to individual network members. Below is a non-exhaustive list of broadcasting models:

- Models with constraints placed on the duration of each call:
    - Constant model, where the time needed to send a message from one node to another remains constant regardless of the size of the message (Fraigniaud & Lazard, 1994).
    - Linear model, where the time required to communicate a message between two neighboring nodes is a linear function of the size of the message (Beauquier, Perennes, & Delmas, 2001; Fraigniaud & Lazard, 1994).
- Models with different directions of the calls: telephone and telegraph communication models (Hedetniemi, Hedetniemi, & Liestman, 1988).
- Single-port or multi-port models discussed by Fraigniaud and Lazard (1994) and Harutyunyan and Liestman (2001b).
- Models with long-distance calls: open-line and open-path models (Farley, 1980b).

- Multiple message broadcasting (Bar-Noy & Kipnis, 1994; Bruck, Cypher, & Ho, 1992; Chinn, Hedetniemi, & Mitchell, 1979; Farley, 1980a).
- Multiple originator broadcasting (Farley, 1980b; Farley & Proskurowski, 1981a; Hedetniemi & Hedetniemi, 1979; Liestman, Richards, & Stacho, 2009).
- Fault-tolerant broadcasting (Ahlswede, Gargano, Haroutunian, & Khachatrian, 1996; Bienstock, 1988; Liestman, 1985; Pelc, 1996).
- Vertex-disjoint and edge-disjoint path models (Farley, 2004; Fraigniaud, 2001a, 2001b).
- Radio broadcasting and conference broadcasting (Alon, Bar-Noy, Linial, & Peleg, 1991; Richards & Liestman, 1988).
- Weighted-vertex broadcasting (Harutyunyan & Kamali, 2008a, 2008b, 2010, 2017)
- Bounded-call broadcasting discussed by Farley and Proskurowski (1994).
- Universal lists broadcasting discussed by Diks and Pelc (1996).
- Messy broadcasting discussed by Ahlswede, Haroutunian, and Khachatrian (1994).

For a more detailed introduction to broadcasting (and gossiping) models, we recommend the reader refers to (Hedetniemi et al., 1988).

In this study, we follow the definitions of the single-port, constant time model (Hedetniemi et al., 1988), known as *classical broadcasting*, in which a node can communicate with a single neighbor at a time, and the transmitting time is constant. Of course, in real-life networks, message spreading takes place very differently from all these models, especially from the classical model. However, from the perspective of research, these constraints help us understand the nature of the phenomenon and achieve theoretical results that can be applied to real-life networks.

2.1 Classical broadcasting

In the context of broadcasting, the network is modeled as an undirected graph $G = (V, E)$, where the vertex set $V(G)$ and the edge set $E(G)$ represent the nodes and the interconnection links

of the network, respectively. Usually, the graph is assumed to be connected. The classical model follows the below-mentioned basic assumptions:

(1) The broadcasting process is split into discrete time units.

(2) The only vertex that has the message at the first time unit is called *originator*.

(3) In each time unit, an informed vertex (*sender*, a vertex that has the message) can *call* at most one uninformed neighbor (*receiver*).

(4) During one time unit, all calls are performed in parallel.

(5) The process halts as soon as all the vertices in the graph are informed.

We can represent each call in this process as an ordered pair of two vertices (u, v), where u is the sender and v is the receiver.

Definition 2.1.1. ***Broadcast scheme*** *is the order of calls made by each vertex during a broadcasting process and can be represented as a sequence $(C_1, C_2, ..., C_t)$, where C_i is the set of calls performed in time unit i.*

An informed vertex v is considered *idle* in time unit t if v does not make any calls in time t. Given that every vertex, other than the originator, can be informed by exactly one vertex, the broadcast scheme forms a directed spanning tree (referred to as *broadcast tree*) rooted at the originator. We are also free to omit the direction of each call in the broadcast tree. Note that if multiple vertices independently inform the same vertex in the same time unit, one of them can be selected as a sender to construct a broadcast tree.

Definition 2.1.2. *The* ***broadcast time*** *for a vertex v in a given graph G, denoted by $b(v, G)$, is the minimum number of time units required for broadcasting in G if v is the originator. The broadcast time for a given graph G is the maximum broadcast time from any originator in G, or formally $b(G) = \max_{v \in V(G)}\{b(v, G)\}$.*

A broadcast scheme for an originator v that uses $b(v, G)$ time units is called optimal broadcast scheme and is denoted by $\mathcal{S}(v, G)$. Obviously, by the assumption (3), the number of informed

vertices after each time unit can be at most doubled. Meaning, in general, the number of informed vertices after time unit i is upper bounded by 2^i. Therefore, it is easy to see that $b(v, G) \geq \lceil \log n \rceil$, where n is the number of vertices in G. This implies that $b(G) \geq \lceil \log n \rceil$.

Generally, one can distinguish two main research directions on broadcasting in graphs.

(1) *Minimum broadcast graph problem:* The goal is to design interconnection networks where message broadcasting finishes as quickly as possible, regardless of the originator.

(2) *Minimum broadcast time problem:* The goal is to find the minimum broadcast time of a network or a specific node.

For a more detailed introduction to broadcasting, we refer the reader to (Fraigniaud & Lazard, 1994; Harutyunyan, Liestman, Peters, & Richards, 2013; Hedetniemi et al., 1988; Hromkovič, Klasing, Monien, & Peine, 1996).

2.2 Minimum broadcast graph problem

Definition 2.2.1. *A graph G on n vertices is called **broadcast graph** if $b(G) = \lceil \log n \rceil$. A broadcast graph G is referred to as a **minimum broadcast graph** (mbg) on n vertices if there does not exist any other broadcast graph G' of the same order as G, such that $|E(G')| < |E(G)|$. The number of edges in a minimum broadcast graph is called **broadcast function** and is denoted by $B(n)$.*

In other words, a broadcast graph is the topology structure supporting optimal broadcasting, and a minimum broadcast graph has the lowest cost, in terms of the number of edges, among such graphs. For a given integer n, the goal of the minimum broadcast graph problem is to find a minimum broadcast graph on n vertices or determine the value of the broadcast function $B(n)$.

The study of the minimum broadcast graph problem has a decades-long history. Minimum broadcast graphs and the broadcast function were first introduced by Farley, Hedetniemi, Mitchell, and Proskurowski (1979). In the same paper, the authors proved that hypercubes are minimum broadcast graphs on 2^k vertices. Currently, there are three known non-isomorphic infinite constructions of minimum broadcast graphs. Hypercubes (Farley et al., 1979), Knödel graphs (Knödel, 1975), and recursive circulant graphs (Park & Chwa, 1994) are all shown to be non-isomorphic

minimum broadcast graphs on 2^k vertices. Knödel graphs are also shown to be minimum broadcast graphs on $n = 2^k - 2$ vertices by Khachatrian and Harutounian (1990) and Dinneen, Fellows, and Faber (1991), independently.

Other than the above-mentioned graph families, minimum broadcast graphs or $B(n)$ are known for some specific values of n.

- $n \leq 16$ (Farley, 1979)
- $n = 17$ (Mitchell & Hedetniemi, 1980)
- $n = 18, 19$ (Bermond, Hell, Liestman, & Peters, 1992b; Xiao & Wang, 1988)
- $n = 20, 21, 22$ (Maheo & Saclé, 1994),
- $n = 26$ (Saclé, 1996; Zhou & Zhang, 2001)
- $n = 27, 28, 29$ (Saclé, 1996)
- $n = 30$ (Bermond, Hell, Liestman, & Peters, 1992b)
- $n = 58, 59, 61$ (Saclé, 1996)

The values of $B(n)$ are also known for some $n = 2^k - 1$, such as

- $n = 31$ (Bermond, Hell, Liestman, & Peters, 1992b)
- $n = 63$ (Labahn, 1994)
- $n = 127$ (Harutyunyan, 2008)
- $n = 1023, 4095$ (Shao, 2006)

The values of $B(n)$ for $n = 23$, 24, and 25 remain unknown.

Even though the minimum broadcast graph problem is widely believed to be "not easy", formally, the NP-hardness of this problem is not proven.

Despite the direct relevance of the minimum broadcast graph problem to the research area of this study, we are not going to focus on minimum broadcast graphs any further.

2.3 Minimum broadcast time problem

The general *broadcast time* decision problem is formally defined as follows:

> **Minimum Broadcast Time (MBT) from (Garey & Johnson, 1983) Problem [ND49]**:
>
> *Given a graph $G = (V, E)$ with a subset $V_0 \subseteq V$, and a positive integer k. Can a message be "broadcast" from the base set V_0 to all other vertices in time k, i.e., is there a sequence $V_0, E_1, V_1, E_2, V_2, \cdots, E_k, V_k$ such that each $V_i \subseteq V$, each $E_i \subseteq E$ $(1 \leq i \leq k)$, $V_k = V$, and, for $1 \leq i \leq k$, (1) each edge in E_i has exactly one endpoint in V_{i-1}, (2) no two edges in E_i share a common endpoint, and (3) $V_i = V_{i-1} \cup \{v : \{u, v\} \in E_i\}$?*

Here k is the total broadcast time, V_i is the set of informed vertices at round i, and E_i is the set of edges used for placing calls at round i. It is obvious that when $|V_0| = 1$ then this problem becomes the single source broadcast problem of determining $b(v, G)$ for an arbitrary vertex v in an arbitrary graph G.

The study of broadcasting can be traced back to 1977 when Slater, Cockayne, and Hedetniemi studied the problem of finding the broadcast time of a given graph (according to Hedetniemi et al. (1988)). Slater, Cockayne, and Hedetniemi (1981) also defined the broadcast center, proposed a linear algorithm to find the broadcast center of a tree, and devised a linear algorithm to find the broadcast time of an arbitrary tree using its broadcast center. The *broadcast center* refers to the set of vertices whose broadcast time is minimum in the given graph. In the same paper, the authors also proved that the decision problem is NP-complete, which was previously mentioned by Garey and Johnson (1983). The NP-hardness proof used a reduction from the *three-dimensional matching* (3DM) problem.

The problem of finding $b(v, G)$ and $b(G)$ are both NP-hard for arbitrary graphs and originators (Garey & Johnson, 1983; Slater, Cockayne, & Hedetniemi, 1981). Later, Middendorf (1993) showed by a reduction from 1-in-3-SAT (*one-in-three* SAT), that the minimum broadcast time problem remains NP-hard even for 3-regular planar graphs. A graph is said to be d-regular if all vertices have a degree of exactly d. The problem is also NP-hard for other restricted graph families, such as bounded degree graphs (Dinneen, 1994).

Concerning the minimum broadcast time problem, like any other NP-hard problem, there are three usual research directions:

- designing **exact algorithms** for restricted instances,
- devising **heuristics** that typically focus on empirical results,
- finding **approximation algorithms** with a theoretically proven guarantee on the worst-case scenario outcome.

2.3.1 Exact algorithms

Firstly, some research was dedicated to formulating the optimization version of the problem and solving it optimally. Dynamic programming approach was suggested by Scheuermann and Wu (1984) and integer linear programming approach was proposed by de Sousa et al. (2018) and Ivanova (2019).

There is a very limited number of graph families for which an exact algorithm with polynomial time complexity is known for the broadcast time problem.

- **Trees**. As mentioned previously, the linear time algorithm for the broadcast time problem in trees was introduced by Slater et al. (1981). One more linear time solution for the same problem was shown by Proskurowski (1981), which finds the broadcast time of a vertex in a given tree without using the broadcast center.
- **Grids and Tori** (Farley & Hedetniemi, 1978),
- **Cube Connected Cycles** (Liestman & Peters, 1988),
- **Shuffle Exchange** (Hromkovič, Jeschke, & Monien, 1993),
- **Unicyclic graphs** (graphs containing a single cycle) graphs (Harutyunyan & Maraachlian, 2007, 2008),
- **Tree of cycles** (Harutyunyan & Maraachlian, 2009b),
- **Necklace graphs** (Harutyunyan, Laza, & Maraachlian, 2009),

- **Tree of cliques** (Maraachlian, 2010),
- **Harary-like graphs** (Bhabak, Harutyunyan, & Kropf, 2017; Bhabak, Harutyunyan, & Tanna, 2014),
- ***k*-cacti** with constant k (k restricted cactus graphs) (Čevnik & Žerovnik, 2017),
- **Fully connected trees** (Gholami, Harutyunyan, & Maraachlian, 2023).

2.3.2 Heuristics

One of the most important characteristics of the broadcasting process is the fact that in each time unit of an optimal broadcast scheme, the edges that are used to transmit the message form a maximum matching between informed and uninformed vertices. Given a graph $G = (V, E)$, let $b(S, G)$ denote the minimum time required to pass the message from the informed set of vertices S to all the other vertices in G. Scheuermann and Wu (1984) introduced the following recurrent relation on $b(S, G)$.

$$b(S, G) = 1 + \min\{b(S \cup M(S), G) \mid M \text{ is a maximum matching between } S \text{ and } V \backslash S\}$$

A backtracking algorithm with exponential run-time that is based on this relation was introduced by Scheuermann and Edelberg (1981). However, the authors did not provide a complexity analysis of the algorithm. Performing such an analysis would be challenging, as the original implementation of the algorithm uses several optimization rules to reduce the run time of the algorithm. Based on the above-mentioned relation, Scheuermann and Wu (1984) also devised three heuristic algorithms for the broadcast time problem. Instead of choosing the maximum matching, all three algorithms use some predefined rules for matching selection.

Harutyunyan and Shao (2006) proposed a heuristic tree-based algorithm (*TBA*) to solve the problem of finding the optimal broadcast scheme of a given graph. They define the *bright border* of the graph, as a subset of vertices that contains informed vertices which have uninformed neighbors. They construct trees rooted at the bright border that contain uninformed vertices. Each uninformed vertex is classified into a tree, based on its distance from other nodes in the bright border, and

is assigned an estimated weight. The algorithm aims to find a specific-weight matching between vertices on different sides of the bright border in each round. The heuristic achieves the running time of $\mathcal{O}(R \cdot |E|)$, where R is the broadcast time returned by the algorithm.

Another heuristic algorithm (*Deep Heuristic*) for finding a good broadcast scheme is introduced by Harutyunyan, Hovhannisyan, and Magithiya (2022). The algorithm has a pattern similar to the TBA and differs by the approach they use for finding the best border crossing. The algorithm is devised based on the disadvantages of other existing heuristics. Specifically, the paper aims to design an algorithm with improved behavior compared to other algorithms (particularly the TBA) in several key scenarios. Comparing the algorithm with other existing alternatives, it was concluded that Deep Heuristic is well-suitable for graphs where most vertices have a high degree, resulting in a higher density of the graph. Deep Heuristic has the complexity of $\mathcal{O}(|E| + |V| \log |V|)$, which is lower than that of many other existing heuristics.

For more information about broadcasting heuristics, we refer the reader to (Albin, Kottegoda, & Poggi-Corradini, 2020; de Sousa et al., 2018, 2020; Fraigniaud & Vial, 1997, 1999; Gholami & Harutyunyan, 2022c; Harutyunyan & Jimborean, 2014; Harutyunyan & Shao, 2006).

2.3.3 Approximation algorithms

Since exact algorithms for solving this problem are not feasible for large networks, heuristic algorithms have been designed to overcome the performance bottleneck. However, there may always be network instances for which any heuristic algorithm will exhibit a significant error. Thus, it is important to devise algorithms that guarantee a limited approximation ratio. It is easy to see that a random broadcasting schedule maintains $\mathcal{O}(\frac{n-1}{\log n})$ approximation ratio. Kortsarz and Peleg (1995) were one of the first to introduce better approximation algorithms. Two main results discussed in the paper are an $\mathcal{O}(\sqrt{n})$ additive approximation algorithm on arbitrary networks in the classical model, and an $\mathcal{O}(\frac{\log n}{\log \log n})$ multiplicative approximation algorithm for broadcasting in the open-path model[1]. The latter is used to derive a broadcasting algorithm for random networks (in the classical model) that with high probability provides $\mathcal{O}(\frac{\log n}{\log \log n})$ approximation ratio.

[1]The *open-path* model assumes calls can be carried via arbitrary long paths in the network. However, all paths used for placing calls in the same round are vertex-disjoint. The *open-line* is similar to the open-path model, except that paths used in the same round need to be edge-disjoint.

Ravi (1994) also investigates the minimum broadcast time problem under the classical model. The keystone of the research is the relation between the minimum broadcast time problem and the poise of a graph[1]. In the paper, an algorithm for finding the spanning tree with approximately minimum poise is used to derive an $\mathcal{O}(\frac{\log^2 n}{\log\log n})$-approximation algorithm for the minimum broadcast time problem of a n-node graph.

Further improvement on the approximation ratio was introduced by Bar-Noy, Guha, Naor, and Schieber (1998). In that paper, the authors mainly investigate a more general *multicast model*, where nodes can have different sending (switching) and receiving times, edges can have different delays, and the message needs to be passed to only a subset of nodes. The nodes that need to be informed are called *terminal nodes*. Assuming that U is the set of terminal nodes and $|U| = k$, the authors devise an approximation algorithm with a multiplicative approximation ratio of $\mathcal{O}(\log k)$ (hence, $\mathcal{O}(\log n)$-approximation for broadcasting).

Current best approximation ratios for the classical broadcast and multicast problems are introduced by Elkin and Kortsarz (2006). The authors devise an approximation algorithm for the classical multicast problem with an approximation ratio of $\mathcal{O}(\frac{\log k}{\log\log k})$, thus providing $\mathcal{O}(\frac{\log n}{\log\log n})$ approximation ratio for the classical broadcast model. In fact, if b^* is the optimum broadcast time, the algorithm guarantees an approximation ratio of $\mathcal{O}(\frac{\log k}{\log b^*})$. A ratio of $\mathcal{O}(\frac{\log k}{\log\log k})$ follows from the fact that $b^* \geq \log k$.

2.3.4 Hardness of approximation

The hardness of approximation of the minimum broadcast time problem has always been an intriguing problem for research. Schindelhauer (2000) investigated approximation algorithms for the broadcasting time problem on undirected graphs. The paper mainly considers the classical model for broadcasting. The author shows that there is no efficient approximation algorithm for the single-source broadcast time problem. Formally, it is NP-hard to distinguish between graphs $G = (V, E)$ with broadcasting time smaller than $b \in \Theta(\sqrt{|V|})$ and larger than $(\frac{57}{56} - \epsilon)b$ for any $\epsilon > 0$.

[1]The *poise* $P(T)$ of a tree T, is the sum of its diameter and its maximum degree. Whereas, the poise $P(G)$ of a graph G, is the minimum poise of all its spanning trees.

A similar hardness of approximation was proved by Bar-Noy et al. (1998) for the more general multicast model, with an inapproximability lower bound of $3-\epsilon$ for any $\epsilon > 0$. In the multicast model considered, only a subset of the network nodes needs to be informed, and delays, as well as switching times, can be arbitrary. The authors prove the inapproximability using a reduction from the set-cover problem.

The current best inapproximability lower bound was proved by Elkin and Kortsarz (2005a). For the broadcast time problem on undirected graphs, they show that it is NP-hard to maintain a $3-\epsilon$ approximation ratio for any $\epsilon > 0$.

2.4 Broadcasting in different families of graphs

This section presents several widely used graph families, their properties, and classical broadcast times.

2.4.1 Path graph P_n

A path P_n is a graph on $n \geq 1$ vertices, $V = \{1, \cdots, n\}$, and the following set of edges: $E = \{(i, i+1) | 1 \leq i \leq n-1\}$. A path P_n has $n-1$ edges, $D(P_n) = n-1$, and $\Delta(P_n) = 2$ when $n \geq 3$. The first and the last vertices have a degree of 1 and all other vertices have a degree of 2. The vertex with the minimum broadcast time in a path graph is the midpoint (either of the midpoints if n is even), denoted by u, with $b(u, P_n) = \lceil \frac{n}{2} \rceil$ (when $n \geq 2$). However, $b(P_n) = n-1$. Path graphs have the maximum sequential delay of broadcasting. Figure 2.1 demonstrates P_6.

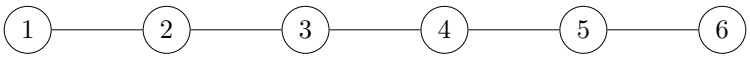


Figure 2.1: Path graph with $n = 6$

2.4.2 Cycle (Ring) C_n

A cycle C_n is a graph on $n \geq 3$ vertices, $V = \{1, \cdots, n\}$, and the edge set $E = \{(i, i+1) | 1 \leq i \leq n-1\} \cup \{(1, n)\}$. A cycle C_n has n edges, is a 2-regular graph, $D(C_n) = \lfloor \frac{n}{2} \rfloor$, and $b(C_n) = \lceil \frac{n}{2} \rceil$. Figure 2.2 demonstrates C_5.

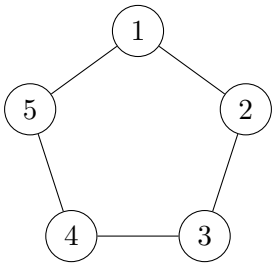


Figure 2.2: Cycle with $n = 5$

2.4.3 Star graph S_n

A star S_n is a graph on $n \geq 1$ vertices, $V = \{1, \cdots, n\}$, and the following set of edges: $E = \{(1, i) | 1 < i \leq n\}$. A star S_n has $n - 1$ edges, $\Delta(S_n) = n - 1$, $b(S_n) = n - 1$, and $D(S_n) = 2$ when $n \geq 3$. Star graphs have the maximum parallel delay of broadcasting. Figure 2.3 demonstrates S_6.

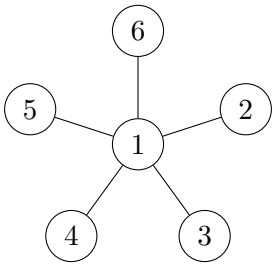


Figure 2.3: Star graph with $n = 6$

2.4.4 Complete graph K_n

A complete graph (clique) K_n is a graph on $n \geq 1$ vertices, $V = \{1, \cdots, n\}$, and all possible edges. A complete graph K_n has $\binom{n}{2} = \frac{n(n-1)}{2}$ edges, is an $(n-1)$-regular graph, $D(K_n) = 1$, and $b(K_n) = \lceil \log n \rceil$. Figure 2.4 demonstrates K_5.

2.4.5 Complete bipartite graph $K_{m,n}$

A complete bipartite graph $K_{m,n}$ consists of two partitions; bipartition P_1 with $m \geq 1$ vertices and bipartition P_2 with $n \geq 1$ vertices. There are no edges between vertices from the same partition. For any $u \in P_1$ and $v \in P_2$, $(u, v) \in E(K_{m,n})$. Also, $deg(u) = n$, and $deg(v) = m$. Figure 2.5 demonstrates $K_{3,4}$. The broadcast time from any originator v in a $K_{m,n}$ is given below:

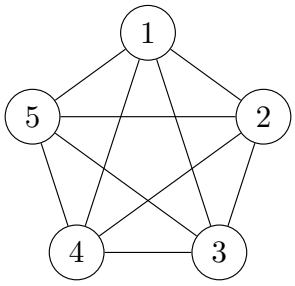


Figure 2.4: Complete graph with $n = 5$

- If $\lceil \log n \rceil = \lceil \log m \rceil$,

$$b(v, K_{m,n}) = 1 + \lceil \log m \rceil$$

- If $\lceil \log n \rceil > \lceil \log m \rceil$,

$$b(v, K_{m,n}) = 1 + \lceil \log m \rceil + \left\lceil \frac{n - 2^{\lceil \log m \rceil}}{m} \right\rceil$$

- If $\lceil \log n \rceil < \lceil \log m \rceil$,

$$b(v, K_{m,n}) = 1 + \lceil \log n \rceil + \left\lceil \frac{m - 2^{\lceil \log n \rceil}}{n} \right\rceil$$

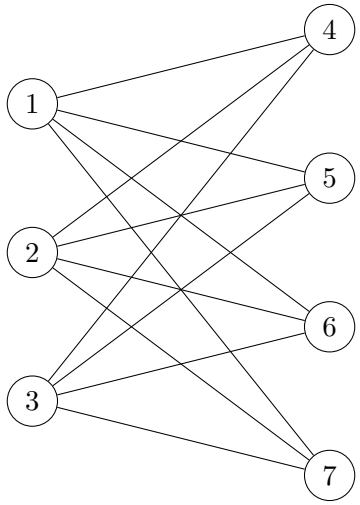


Figure 2.5: Complete bipartite graph with $m = 3, n = 4$

2.4.6 Fork graph $F_{n,k}$

A Fork $F_{n,k}$, where $1 \leq k < n$, has n vertices and comprises a path on $n-k$ vertices and a star on $k+1$, such that the center of the star is one of the endpoints of the path. A fork $F_{n,k}$ has $n-1$ edges, $D(F_{n,k}) = n-k$, and $b(F_{n,k}) = n-1$. Figure 2.6 demonstrates $F_{9,5}$.

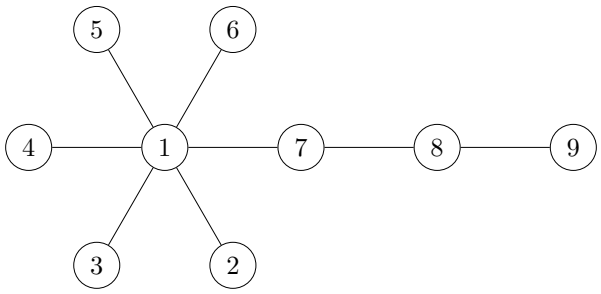


Figure 2.6: Fork graph with $n = 9$ and $k = 5$

2.4.7 Wheel graph W_n

A wheel W_n is a graph on $n \geq 4$ vertices, $V = \{1, \cdots, n\}$, and the following set of edges: $E = \{(1,i)|1 < i \leq n\} \cup \{(i,i+1)|2 \leq i \leq n-1\} \cup \{(2,n)\}$. A wheel W_n has $2n-1$ edges, $D(W_n) = 2$ if $n > 4$. Whereas, $\Delta(W_n) = n-1$, and $b(W_n) = \left\lceil \frac{\sqrt{4n-3}+1}{2} \right\rceil$. Figure 2.7 demonstrates W_6.

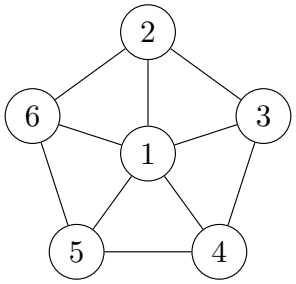


Figure 2.7: Wheel graph with $n = 6$

2.4.8 Grid $G_{m\times n}$

A two-dimensional Grid (Lattice), $G_{m\times n}$, is a graph on $m \times n$ vertices, where $n \geq 1$ and $m \geq 1$. Each of the vertices is on integer coordinates of a Cartesian plane and there is an edge between vertices if they have Euclidean distance one. A grid $G_{m\times n}$ has $(m-1)n + (n-1)m =$

$2mn - (m + n)$ edges, $D(G_{m\times n}) = m + n - 2$, $b(G_{m\times n}) = m + n - 2$ (Farley & Hedetniemi, 1978), and $\Delta(G_{m\times n}) = 4$ when $n \geq 3$ and $m \geq 3$. Figure 2.8 demonstrates $G_{5\times 3}$.

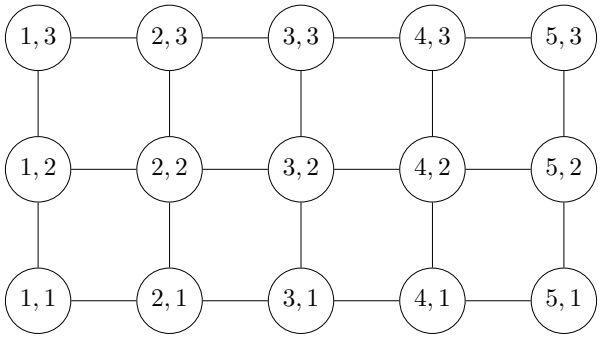


Figure 2.8: Grid with $m = 5, n = 3$

2.4.9 Torus $T_{m\times n}$

A two-dimensional Toroidal Grid (Torus), $T_{m\times n}$, is a graph on $m \times n$ vertices, where $n \geq 2$ and $m \geq 2$. It includes all edges of the grid and edges $((i, 1), (i, n))$ and $((1, j), (m, j))$, $\forall i, j$ such that $1 \leq i \leq m, 1 \leq j \leq n$. Tori have $2mn$ edges, are 4-regular graphs, $D(T_{m\times n}) = \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor$. If both m and n are even, then $b(T_{m\times n}) = \frac{m+n}{2}$, otherwise, $b(T_{m\times n}) = \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor + 1$ (Farley & Hedetniemi, 1978). Figure 2.9 demonstrates $T_{5\times 3}$.

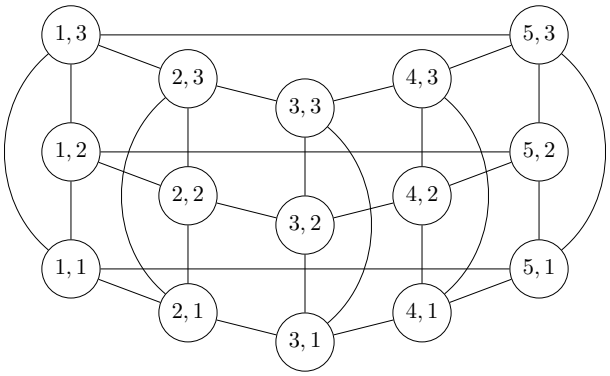


Figure 2.9: Torus with $m = 5, n = 3$

2.4.10 Hypercube Q_d

The d-dimensional hypercube, Q_d, is a graph on 2^d vertices. Each vertex is a d-bit binary string, and two vertices are adjacent if and only if they differ at only one bit. Alternatively, Q_d

can be defined recursively as a union of two copies of Q_{d-1}, where the corresponding vertices in two copies are connected by an edge. A hypercube Q_1 is a single-edge graph on two vertices. A hypercube Q_d has $d \cdot 2^{d-1}$ edges, is d-regular, bipartite, and $D(Q_d) = d$. Hypercube is one of the infinite families of broadcast graphs: $b(Q_d) = \lceil \log |Q_d| \rceil = d$. Figure 2.10 demonstrates Q_3.

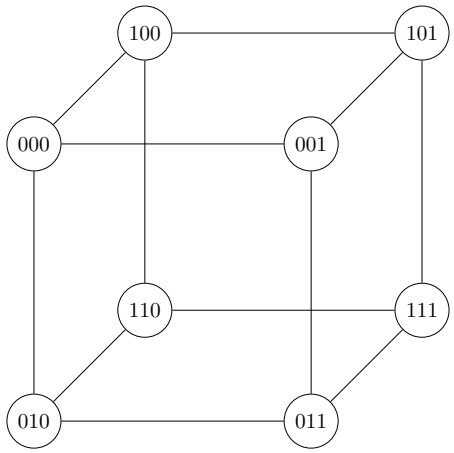


Figure 2.10: Hypercube with $d = 3$

2.4.11 Complete k-ary tree $T_{k,h}$

A k-ary tree is a rooted tree in which the number of children of each internal vertex is k. The degree of the root is k, the degrees of internal vertices are $k + 1$, and the leaves' degrees are 1. A complete k-ary tree of height h, (denoted by $T_{k,h}$), is a rooted k-ary tree in which all leaves are on the same level h. A tree $T_{k,h}$ has $\frac{k^{h+1}-1}{k-1}$ vertices, $D(T_{k,h}) = 2h$, $\Delta(T_{k,h}) = k + 1$, and $b(T_{k,h}) = kh + h - 1$. Figure 2.11 demonstrates $T_{5,2}$.

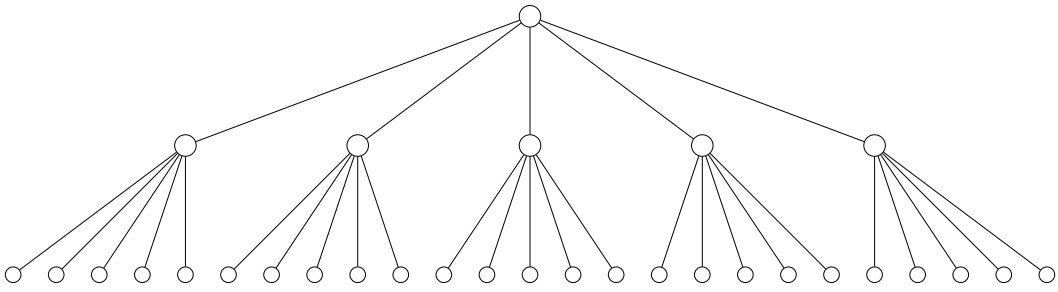


Figure 2.11: k-ary tree with $k = 5, h = 2$

2.4.12 Binomial tree BT_d

A binomial tree of dimension d is a rooted tree on 2^d vertices with a recursive definition. The binomial tree of dimension 0 (BT_0) is a single vertex. A binomial tree BT_d comprises two copies of BT_{d-1}, where the roots are connected by an edge. One of the roots is selected to be the root r of the new tree. A tree BT_d has $2^d - 1$ edges, $D(BT_d) = 2d - 1$, $\Delta(BT_d) = d$, $b(r, BT_d) = d = \log |BT_d|$, and $b(BT_d) = 2d - 1$. Figure 2.12 demonstrates BT_4.

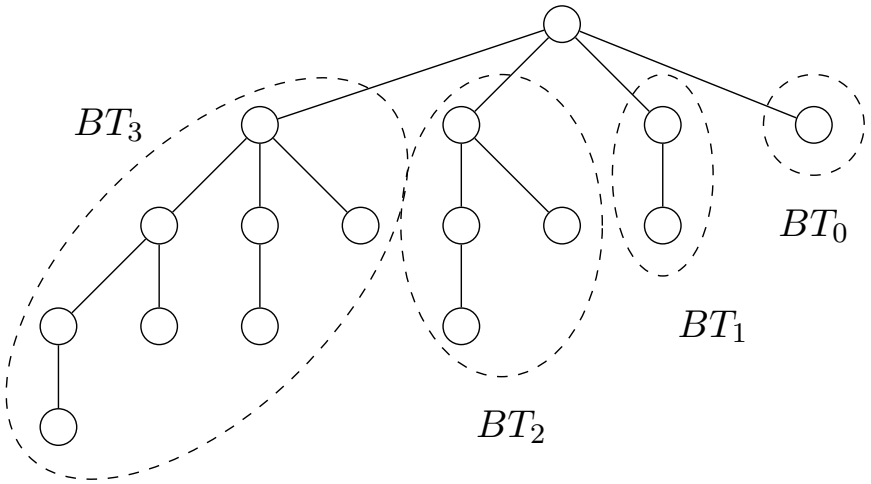


Figure 2.12: Binomial tree with $d = 4$

2.4.13 Recursive circulant graph $G(n, d)$

The recursive circulant graph, defined by Park and Chwa (1994), has $n = cd^m$ vertices labeled from 0 to $n - 1$, where $d \geq 2$, $m \geq 1$, and $1 \leq c < d$. The edge set is defined as $E = \{(v, w) |$ there exists $i, 0 \leq i \leq \lceil \log_d n \rceil - 1$, such that $v + d^i \equiv w \mod n\}$.

Interestingly, $G(2^m, 4)$ has the same number of nodes, edges ($m \cdot 2^{m-1}$), and the broadcast time ($b(G(2^m, 4)) = \lceil \log 2^m \rceil = m$) as the hypercube Q_m. A recursive circulant graph $G(n, d)$ is $m = \lceil \log n \rceil$-regular, and $D(G(n, d)) = \lceil \frac{3m-1}{4} \rceil$. Figure 2.13 demonstrates $G(2^3, 4)$.

2.4.14 Knödel graph $KG_{\Delta,n}$

The Knödel graph $KG_{\Delta,n}$ is defined as a graph on n vertices labeled from 0 to $n-1$, where $n \geq 6$ and is even (Bermond, Harutyunyan, Liestman, & Pérennes, 1997; Khachatrian & Harutounian,

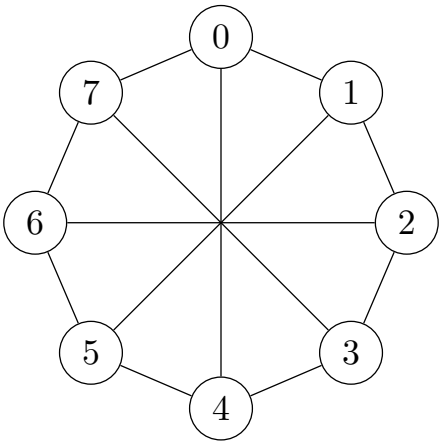


Figure 2.13: Recursive Circulant graph with $n = 8, d = 4$

1990; Knödel, 1975). The edge set is defined as $E = \{(x, y) | x + y \equiv 2^{\Delta} - 1 \mod n\}$, where $1 \leq \Delta \leq \lfloor \log n \rfloor$. The Knödel graph $KG_{\Delta,2^{\Delta}}$ is a Δ-regular graph with diameter $\lceil \frac{\Delta+2}{2} \rceil$.

As mentioned in Section 2.2.1, the family of $KG_{\Delta,2^{\Delta}}$ is one of the three known infinite families of minimum broadcast graphs. This means that $b(KG_{\Delta,2^{\Delta}}) = \Delta = \log n$. Additionally, it is known that $b(KG_{\Delta,2^{\Delta}-2}) = \Delta$ ($\Delta \geq 2$) (Dinneen, Fellows, & Faber, 1991). Figure 2.14 demonstrates $KG_{3,14}$.

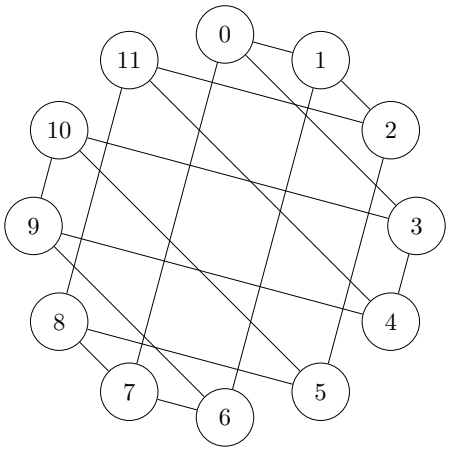


Figure 2.14: Knödel graph with $\Delta = 3, n = 14$

Chapter 3

Split Graphs

In this chapter, our contributions concerning split graphs are presented. We will devise an approximation algorithm for the broadcast time problem in split graphs that guarantees an approximation ratio of 2. We will also analyze the characteristics of an optimal broadcast scheme in split graphs. After that, we will suggest a heuristic algorithm for the same problem and an empirical analysis of the proposed heuristic. Lastly, we will introduce an algorithm for the broadcast time problem in arbitrary graphs with a known partitioning.

3.1 Introduction

A split graph is a graph in which the vertices can be partitioned into a clique and an independent set (Merris, 2003).

Definition 3.1.1. *A graph $SP_{n,m}$ is a **split graph** if the vertex set can be partitioned into:*

- *A clique K of size n,*
- *And an independent set I of size m.*

Figure 3.1 portrays an example of a split graph $SP_{3,3}$.

Split graphs were first studied by Foldes and Hammer (1977) and independently introduced by Tyshkevich and Chernyak (1979). An edge of a graph is a chord of a cycle if it joins two nodes of the cycle but is not itself in the cycle. A graph is chordal if and only if every cycle of length greater

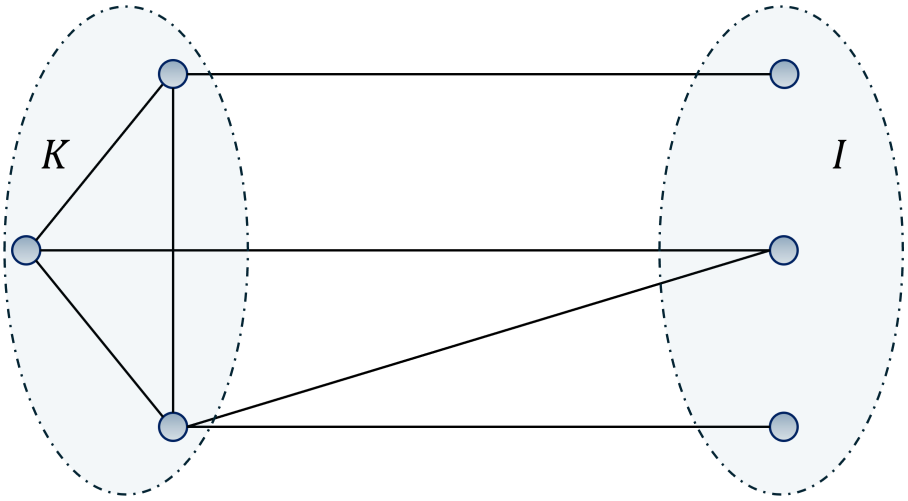


Figure 3.1: Split graph $SP_{3,3}$

than three has a chord. Split graphs are a subclass of chordal graphs and are exactly those chordal graphs whose complement, i.e., the same graph in which edges and nonedges are swapped, is also chordal (Golumbic, 2004). Therefore, all problems which are polynomial-time solvable for chordal graphs are also solvable for split graphs (Garey & Johnson, 1983; Golumbic, 2004). Chordal graphs play a central role in techniques for exploiting sparsity in large semidefinite optimization problems (Zhang & Lavaei, 2018) and in related convex optimization problems involving sparse positive semidefinite matrices. Chordal graph properties are also fundamental to several classical results in combinatorial optimization, linear algebra, statistics, signal processing, machine learning, and nonlinear optimization (Vandenberghe & Andersen, 2015; Zheng, 2019).

Bender, Richmond, and Wormald (1985) showed that in the limit as n goes to infinity, the fraction of n-vertex chordal graphs that are split approaches one. Less formally, they showed that almost all chordal graphs are split graphs, thus making split graphs an important area of research. Moreover, split graphs are widely used as an interconnection network topology. Networks that have an important group of tightly coupled nodes (or in other words the *core* of the network), and several independent nodes that are only connected to the network core are often represented as split graphs. The same structure is possible in dynamic networks, where nodes can join and leave the network by connecting to the physically closest nodes in the core. In terms of social networks, split graphs correspond to the variety of interpersonal and intergroup relations (Belik, 2016). The interaction

between the cliques (socially strong and trusty groups) and the independent sets (fragmented and non-connected groups of people) is naturally represented as a split graph. Different optimization problems were studied in split graphs due to their various important characteristics (Belmonte et al., 2021; Collins & Trenk, 2017a). Some of the problems that are NP-complete in the general case are fairly trivial in split graphs. However, the tasks of finding a Hamiltonian cycle, a Minimum Dominating Set, or a λ-coloring remain NP-complete on split graphs (Bertossi, 1984; Bodlaender, Kloks, Tan, & van Leeuwen, 2000; Müller, 1996). Given all of the above, it is interesting to study the problem of broadcasting in split graphs.

Jansen and Müller (1995) were one of the first to study the minimum broadcast time problem on split graphs. In the paper, the authors prove that it is NP-complete to decide whether the broadcast time from multiple originators in split graphs (and in chordal graphs) is equal to 2. In the same paper, the authors also show that the minimum broadcast time problem from a single originator in chordal graphs is NP-complete. Later, in (Tamura, Tasaki, Sengoku, & Shinoda, 2005), the minimum broadcast and multicast time problems are considered for a subclass of split graphs where the degree of all vertices in the independent set is one. The authors introduce a polynomial-time exact algorithm for that restricted subclass of split graphs.

3.2 Broadcasting approximation in split graphs

Consider a split graph $G = (V, E)$ such that the vertex set can be partitioned into a clique K on n vertices and an independent set I on m vertices. Note that split graphs may have more than one partitioning into a clique and an independent set. Clearly, we can assume that the clique K is maximal, otherwise, we could change the partitioning by adding some vertices from I to K to make it maximal. Given a connected split graph G with a clique partition K, an independent set partition I, and an arbitrary originator $v \in V(G)$ we consider the following two cases.

(1) The originator is from the clique: $v \in K$.

(2) The originator is from the independent set: $v \in I$.

Before proceeding to the actual algorithm we will discuss some results and definitions that will

be helpful in our proof. First, let us introduce some definitions and consider a broadcasting problem with multiple originators.

Definition 3.2.1 (Lin & Lam, 2009)**.** *For a positive integer t, a **t-star-matching** of graph G is a collection of mutually vertex-disjoint subgraphs $K_{1,i}$ of G with $1 \leq i \leq t$ (sub-stars with maximum t leaves (Section 2.4.3)) .*

We extend the idea of a t-star-matching (Definition 3.2.1) to the following concepts.

Definition 3.2.2. *For a positive integer t, a **perfect t-star-matching** of a graph G is a t-star-matching that covers every vertex of graph G.*

Definition 3.2.3. *Let G be a split graph, K be the clique partition of G, and t be a positive integer. A **proper t-star-matching** of a split graph G is a perfect t-star-matching such that any vertex $v \in K$ belongs to the smaller partition of a bipartite subgraph (is a center of a star).*

Note that a proper star-matching may contain stars with no leaves (some clique vertices may not have any incident edges in the matching). Figure 3.2 shows an example of proper 2-star-matching on a $SP_{4,4}$ split graph.

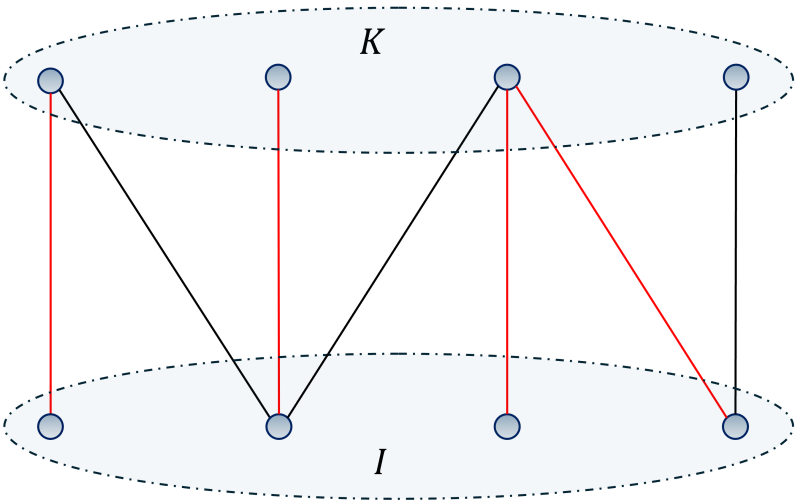


Figure 3.2: Example of a proper 2-star-matching of a $SP_{4,4}$. Red edges form the star-matching.

Lemma 3.2.4. *Consider a split graph $G = (V, E)$ with a partitioning of a maximal clique K and an independent set I, and a set of originators R such that $R = K$. For a positive integer t, there exists a proper t-star-matching of G, if and only if $b(K, G) \leq t$.*

Proof. We will prove this in two parts. First, assume $b(K, G) \leq t$ for an integer $t > 0$. Let S be an optimal broadcast scheme for G originating from K. It is easy to see that in the case of multiple originators, any broadcast scheme forms a directed spanning forest (*broadcast forest*), where each tree in the forest is rooted at one of the originators. Note that any vertex $u \in I$, cannot inform any other vertex after being informed, since all of its neighbors are in K and are initially informed. It follows that any broadcast forest of a split graph has a depth of one. Hence, the broadcast scheme will induce a set of stars, where the center of each star is one of the originators. Moreover, since the broadcasting process takes $b(K, G) \leq t$ time units, none of the vertices in K can have more than t neighbors in the broadcast forest. Thus, the broadcast forest of S is a proper t-star-matching of G.

Now, assume there exists a proper t-star-matching M of G. Consider the broadcast scheme S' where the edges that form M are used to broadcast the message from K to I. Each vertex in K, informs its neighbors in M in an arbitrary order. Let v be the last vertex in K that finished informing its neighbors in the independent set. Since, $deg_M(v) \leq t$, v will inform all of its neighbors by time unit t. Thus, $b(K, G) \leq b_S(K, G) \leq t$. □

By Lemma 3.2.4, if we find the minimum positive integer t, for which there exists a proper t-star-matching, we can claim that $b(K, G) = t$.

3.2.1 Finding a proper star-matching with minimum maxdegree

To find the minimum positive integer t, for which there exists a proper t-star-matching, we pose the following decision problem.

Problem 3.2.1.1. *Proper star-matching problem*

Instance: *(G, K, I, t), where $G = (V, E)$ is a split graph, K is a clique in G, I is an independent set in G, and t is a natural number.*

Output: *"Yes" if there exists a proper t-star-matching for graph G; "No" otherwise.*

Next, we will reduce an instance of the above-defined problem to an instance of a maximum flow problem (Fulkerson & Ford, 1962; Schrijver, 2002).

Problem 3.2.1.2. *Maximum flow*

Instance: *(G, s, q, m), where $G = (V, E)$ is a flow graph, s is the source vertex, q is the sink vertex,*

and m is a natural number.

Output: *"Yes" if the maximum flow for graph G from s to q is greater than or equal to m; "No" otherwise.*

The reduction consists of the steps described in Algorithm 1, and an example of the reduction is presented in Figure 3.3.

Algorithm 1 Reduction Algorithm

Input An instance of the proper star-matching problem (G, K, I, t)
Output An instance of the maximum flow problem (G, s, q, m)

procedure MAXFLOWREDUCTION
    Create a copy G' of graph G
    Remove all edges between a pair of vertices u and v, where $u, v \in K$
    Assign direction from u to v to each edge (u, v), where $u \in K$, $v \in I$
    Assign capacity 1 to each edge (u, v), where $u \in K$, $v \in I$
    Add a source vertex s and connect it by an outgoing edge to every vertex in K. Assign a capacity of t to those edges
    Add a sink vertex q and connect it by an ingoing edge to every vertex in I. Assign a capacity of 1 to those edges
    return $(G', s, q, |I|)$
end procedure

Proposition 3.2.1.3. *There exists a proper t-star-matching for a split graph G, if and only if the maximum flow in graph G' is m, where m is the cardinality of the independent set I.*

Proof. It is easy to see that when there exists a proper t-star-matching, the same edges can be used in graph G' to achieve flow with value m. Similarly, if the maximum flow equals m, the paths used to construct the flow can be used to find a proper star-matching for graph G. Since each vertex in K has an incoming edge capacity of t, it will result in a proper t-star-matching for graph G. □

Thus, we can use the output of the procedure MAXFLOWREDUCTION to devise a procedure to check if there exists a proper t-star-matching of a split graph G for a given positive integer t (Algorithm 2). In Algorithm 2, procedure CHECKMATCHING is used to execute a binary search to find the minimum such value of t in the range $1 \leq t \leq m$ (procedure FINDMATCHING).

The procedure MAXFLOWREDUCTION (Algorithm 1) can be implemented with complexity $\mathcal{O}(n) + \mathcal{O}(m) = \mathcal{O}(|V|)$. The time to find the maximum flow in a graph highly depends on the

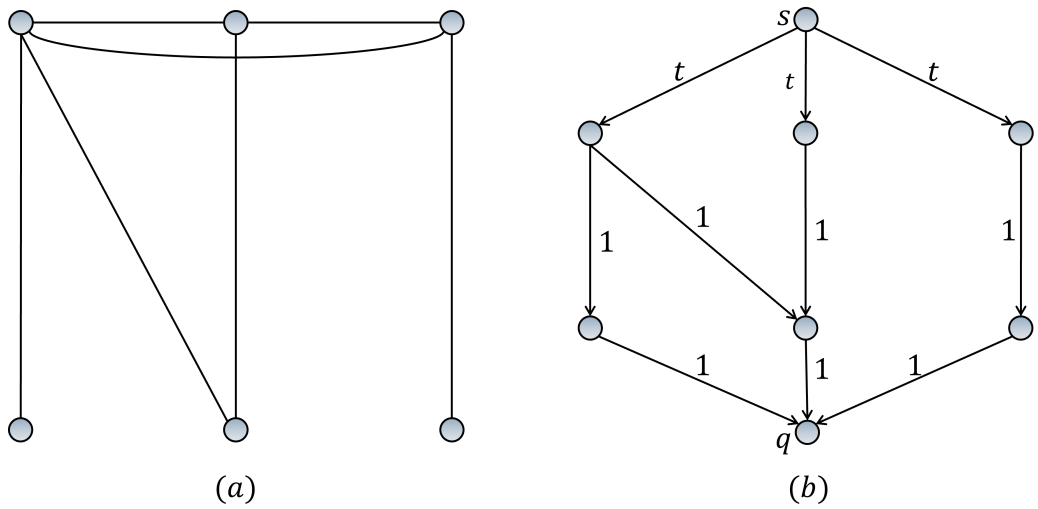


Figure 3.3: Example of a reduction in Algorithm 1. Subfigure (a) shows graph G before the reduction and subfigure (b) shows graph G' after the reduction.

selected algorithm. One of the best algorithms for maximum flow has a complexity of $\mathcal{O}(|E||V|)$ (Orlin, 2013). Thus, considering the binary search time, the overall time complexity of this algorithm is $\mathcal{O}(|E||V| \log |I|)$.

3.2.2 Broadcasting from a vertex in the clique

Consider a split graph $G = (V, E)$ with a clique partition K, an independent set partition I, and an originator $v \in K$. The algorithm consists of three major stages:

(1) Since $v \in K$ and K is a clique (a complete graph), we can broadcast the message from v to all other vertices of K using only vertices in K. This will take $b(v, K) = \lceil \log |K| \rceil = \lceil \log n \rceil$ time units (Section 2.4.4).

(2) After all vertices in K are informed, we should proceed to inform the vertices in I. For that purpose, we find a proper t-star-matching M of G with minimum t using the algorithms in Section 3.2.1.

(3) Lastly, we use the edges that form M to broadcast the message to I. Each vertex in K, informs its neighbors in M in an arbitrary order. This will take $b(K, G) = t$ time units (Lemma 3.2.4).

Algorithm 2 Algorithm for finding a proper star-matching

Input A split graph G, a clique K, an independent set I
Output A proper t-star-matching M for G with minimum t

procedure FINDMATCHING(G)
  for Binary search t on range $[1, |I|]$ **do**
    $C \leftarrow$ CHECKMATCHING(t)
    if $C = NULL$ **then**
      Search in the right half
    else
      $M \leftarrow C$
      Search in the left half
    end if
  end for
  return M
end procedure
procedure CHECKMATCHING(t)
  Use $MaxFlowReduction$ to create a flow graph G'
  Run a maximum flow algorithm on G'
  $M \leftarrow$ the edges used in the maximum flow
  $V(M) \leftarrow$ the maximum flow number
  if $V(M) = |I|$ **then**
    return M
  else
    return $NULL$
  end if
end procedure

Algorithm 3 formally presents the above-described steps. Clearly, after the algorithm halts, all vertices of graph G are informed as the clique is informed in step 2, and the independent set is informed in step 4.

Let $b_{Alg}(v, G)$ be the broadcast time returned by Algorithm 3. It is easy to see that it consists of two components:

- the broadcast time of the clique K from a single originator,
- and the broadcast time in G with the vertices of K as originators.

Hence, for $n = |K|$, the following is evident.

$$b_{Alg}(v, G) = b(K_n) + b(K, G) = \lceil \log n \rceil + t \tag{1}$$

Algorithm 3 Approximation Algorithm

Input A split graph G, a clique K, an independent set I, and an originator $v \in K$
Output A broadcast scheme with time $b_{Alg}(v, G)$ for graph G and the originator v

1: **procedure** BROADCASTINGFROMCLIQUE
2: Broadcast in the clique K starting from the vertex v
3: Find a proper t-star-matching M with minimum t (Algorithm 2)
4: Use the edges of M to broadcast from K to I
5: **end procedure**

Recall that for any graph G, $b(v, G) \geq \lceil \log |V(G)| \rceil$. As in our case $|V(G)| = n + |I| \geq n$, then $b(v, G) \geq \lceil \log n \rceil$. Moreover, it is trivial that the broadcast time from a single originator in the clique cannot be less than the broadcast time where all the vertices in the clique are originators. Hence, $b(v, G) \geq b(K, G) = t$. From the two lower bounds of the minimum broadcast time described above, it follows that:

$$b(v, G) = \frac{b(v, G)}{2} + \frac{b(v, G)}{2} \geq \frac{\lceil \log n \rceil}{2} + \frac{t}{2} \tag{2}$$

Thus,

$$\frac{b_{Alg}(v, G)}{b(v, G)} \leq \frac{\lceil \log n \rceil + t}{\frac{\lceil \log n \rceil + t}{2}} = 2 \tag{3}$$

We can see that the broadcast process of Algorithm 3 is guaranteed to be at most twice as long as the optimal broadcast process.

As steps 2 and 4 of Algorithm 3 have known broadcast times, they have $\mathcal{O}(1)$ time complexity. Thus, the proposed algorithm has a complexity of $\mathcal{O}(|E||V| \log |I|)$, making it a polynomial-time 2-approximation algorithm for the broadcast time problem in split graphs from an originator in the clique.

3.2.3 Broadcasting from a vertex in the independent set

Consider a split graph $G = (V, E)$ with a clique partition K, an independent set partition I, and an originator $v \in I$. The algorithm consists of three major steps:

(1) In the first time unit, v informs an arbitrary neighbor $u \in K$.

(2) A new split graph G' is constructed by copying graph G and removing vertex v.

(3) Starting from the second time unit, the process of broadcasting in G' originating from u is executed following the Algorithm 3.

Algorithm 4 formally presents the steps of broadcasting from an originator in the independent set. After the algorithm halts, all vertices of graph G are informed.

Algorithm 4 Approximation Algorithm

Input A split graph G, a clique K, an independent set I, and an originator $v \in I$
Output A broadcast scheme with time $b_{Alg}(v, G)$ for graph G and the originator v

1: **procedure** BROADCASTINGFROMINDEPENDENTSET
2: Pass the message to an arbitrary neighbor $u \in K$ of v
3: Complete broadcasting in the clique K
4: Copy a graph G' from G by removing the vertex v
5: Find a proper t-star-matching M with minimum t for split graph G' (Algorithm 2)
6: Use the edges of M to broadcast from K to I
7: **end procedure**

Let $b_{Alg}(v, G)$ be the broadcast time generated by Algorithm 4. It is easy to see that it has three components:

- the time to send the message from v to u,
- the broadcast time of the clique K with u as the originator,
- and the broadcast time in G' with the vertices of K as originators.

Hence, we can claim that $b_{Alg}(v, G) = 1 + b(u, K_n) + b(K, G')$, where $n = |K|$.

$$b_{Alg}(v, G) = 1 + b(u, K_n) + b(K, G') = 1 + \lceil \log n \rceil + t \tag{4}$$

Recall that for any graph G, $b(v, G) \geq \lceil \log |V(G)| \rceil$. Since in our case $n = |K|$ and $m = |I|$, then $b(v, G) \geq \lceil \log(n+m) \rceil \geq \lceil \log n \rceil$. Moreover, since in any broadcast scheme, v should inform a vertex in K in the first time unit, the broadcast time from a single originator in the independent set is greater by at least one than the broadcast time where all the vertices in the clique are originators and the rest of the independent set should be informed. Hence, $b(v, G) \geq 1 + b(K, G') = 1 + t$. From the two lower bounds of the minimum broadcast time described above, it follows that:

$$b(v, G) \geq \frac{\lceil \log n \rceil + t + 1}{2} \tag{5}$$

Thus,

$$\frac{b_{Alg}(v,G)}{b(v,G)} \leq \frac{\lceil \log n \rceil + t + 1}{\frac{\lceil \log n \rceil + t + 1}{2}} = 2 \tag{6}$$

Algorithm 4, similar to Algorithm 3, has a time complexity of $\mathcal{O}(|E||V| \log |I|)$, rendering it a polynomial-time 2-approximation algorithm for the broadcast time problem in split graphs from an originator in the independent set.

3.2.4 Tightness of approximation

In this section, we will prove that the approximation ratio of 2 is tight for the approximation algorithms introduced in Sections 3.2.2 and 3.2.3 (Algorithm 3 and 4). For that, we will construct an infinite subfamily of split graphs, for which 2 is a tight approximation ratio when the originator is in the clique or the independent set.

Claim 3.2.4.1. *For every positive integer $0 < \epsilon < 1$, there exists a split graph instance G and an originator v, for which $b_{Alg}(v,G) > (2-\epsilon) \cdot b(v,G)$, where $b_{Alg}(v,G)$ is the broadcast time returned by Algorithm 3 or Algorithm 4 (depending on the originator) and $b(v,G)$ is the minimum broadcast time.*

Proof. Let G be a split graph with a clique partition $K = \{v_1, v_2, \ldots, v_n\}$ and an independent set partition $I = \{w_1, w_2, \ldots, w_m\}$, such that $n = |K| = 2^t + 1$ and $m = |I| = t$ for some positive integer t. Assume vertex $v_1 \in K$ is adjacent to every vertex $w_i \in I$, $1 \leq i \leq m$, and there exist no more edges between the clique and the independent set (Figure 3.4). We will analyze the broadcast time in 2 cases: v_1 is the originator and w_1 is the originator.

Consider the following broadcast scheme S_1 for originator v_1.

(1) Place a call from v_1 to v_2 in the first time unit.

(2) In the next t time units

   (a) v_1 informs its neighbors in I,

   (b) vertices in $K \setminus \{v_1\}$ finish broadcasting in the clique with v_2 as the originator.

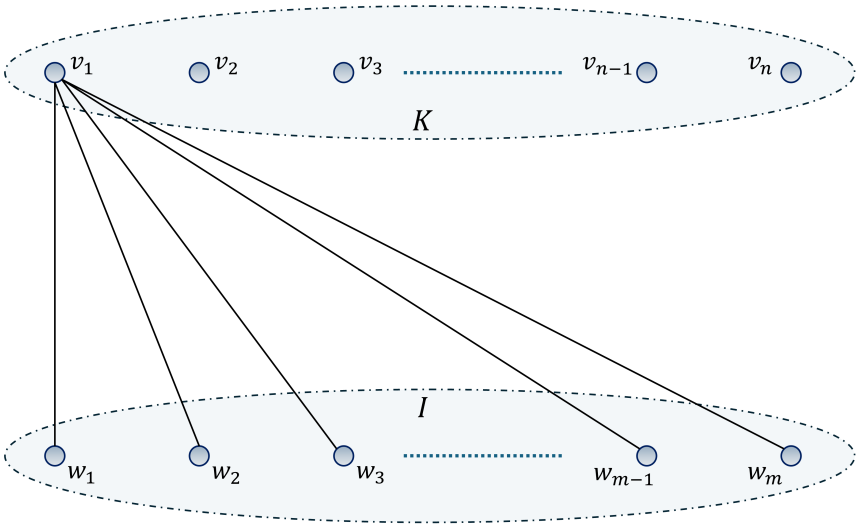


Figure 3.4: Example of a split graph with tight approximation ratio.

Clearly, the broadcast scheme S_1 will have a broadcast time of $b_{S_1}(v_1, G) = t + 1 = \lceil \log n \rceil = b(v_1, G)$, which is the trivial lower bound. Whereas, Algorithm 3 will return a broadcast time $b_{Alg}(v_1, G)$, such that $b_{Alg}(v_1, G) = \lceil \log n \rceil + m = 2t + 1$. Hence, after simple calculations, we can show that for every $0 < \epsilon < 1$, $b_{Alg}(v_1, G) > (2 - \epsilon) \cdot b(v_1, G)$, when $t > \dfrac{1 - \epsilon}{\epsilon}$.

Next, consider the broadcast scheme S_2 for originator w_1.

(1) Place a call from w_1 to v_1 in the first time unit.

(2) Place a call from v_1 to v_2 in the second time unit.

(3) In the next t time units

   (a) v_1 informs its uninformed neighbors in I,

   (b) vertices in $K \setminus \{v_1\}$ finish broadcasting in the clique with v_2 as the originator.

The broadcast scheme S_2 will have a broadcast time of $b_{S_2}(w_1, G) = t + 2 = 1 + \lceil \log n \rceil = b(w_1, G)$, which is again the trivial lower bound. Whereas, Algorithm 4 will return a broadcast time $b_{Alg}(w_1, G) = \lceil \log n \rceil + (m - 1) + 1 = 2t + 1$. Hence, after simple calculations, we can show that for every $0 < \epsilon < 1$, $b_{Alg}(w_1, G) > (2 - \epsilon) \cdot b(w_1, G)$, when $t > \dfrac{3 - 2\epsilon}{\epsilon}$.

□

3.3 Analysis of an optimal broadcast scheme

In the context of this section, we will be focusing on the decision version of the broadcast time problem (MBT). Before introducing the actual algorithm we will discuss some claims and notations that will be used in it.

Lemma 3.3.1. *Let G be a split graph, K be a clique partition of G, and I be an independent set partition of G. There exists an optimal broadcast scheme S for graph G starting from an originator in K, such that every vertex $u \in I$ has no uninformed neighbors after the time unit when u gets informed.*

Proof. In other words, Lemma 3.3.1 claims that there exists an optimal broadcast scheme where no calls are placed from a vertex in the independent set to a vertex in the clique. Let OPT be an optimal broadcast scheme. If OPT does not contain any calls placed from the independent set to the clique, then the lemma is proved. Assume in scheme OPT, a vertex $u \in I$ is informed by a vertex $v \in K$ in time unit i, and in time units $i \leq t_1 \leq t_2 \leq ... \leq t_p$, for some $p > 0$, u informs its uninformed neighbors $w_1, w_2, ..., w_p \in K$, respectively. Then we construct a new broadcast scheme S by copying OPT with the following changes (depicted in Figure 3.5):

- In time unit i, v informs w_1, instead of u, as they are both in the clique. In the rest of the broadcasting process, v continues as in OPT.
- In time unit t_j, vertex w_j passes the message to w_{j+1}, where $1 \leq j \leq p-1$. After time unit t_j, w_j behaves as in OPT.
- In time unit t_p, vertex u is informed by its neighbor w_p. After time unit t_p, all vertices behave as in OPT.

This transformation is visualized in Figure 3.5, where subfigure (a) shows the order of calls in scheme OPT and subfigure (b) shows the order of calls in scheme S. This shuffle will not affect the broadcast time, as after time unit t_p we will have the same set of vertices informed. The same transformation can be applied to any other situation where a call is placed from the independent set to the clique. Hence, the created broadcast scheme S will have a broadcast time equal to that of OPT, thus becoming an optimal broadcast scheme that satisfies the conditions of the lemma. □

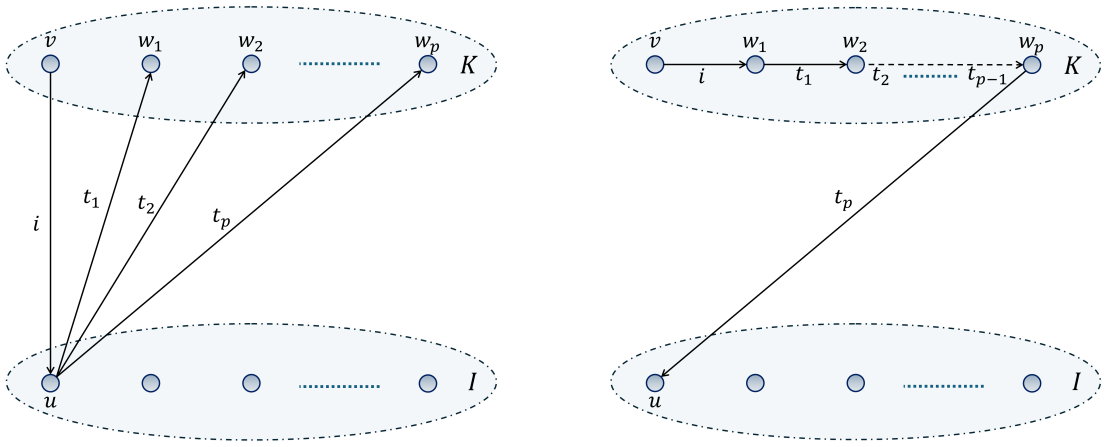


Figure 3.5: Example of a transformation discussed in Lemma 3.3.1.

From Lemma 3.3.1, it is clear that any calls placed from a vertex $v \in K$ to its neighbors in K should be performed before calls towards its neighbors in I.

Lemma 3.3.2. *Let G be a split graph, K be a clique partition of G, and I be an independent set partition of G. There exists an optimal broadcast scheme S originating from a vertex in K and satisfying the conditions of Lemma 3.3.1, such that no vertex $u \in K$ informs a vertex $w \in K$ after a vertex $v \in I$.*

Proof. In other words, vertices in the clique start placing calls to their neighbors in the independent set after they are done informing their neighbors in the clique.

Let $((u, v), (u, w))$ be a pair of calls (pair of directed edges), such that $u, w \in K$ and $v \in I$. We say that $((u, v), (u, w))$ is a *bad pair of calls* in a broadcast scheme S if the call (u, v) was placed before the call (u, w). Let OPT be an optimal broadcast scheme satisfying the conditions of Lemma 3.3.1. It is clear that if OPT contains no bad pair of calls then it satisfies Lemma 3.3.2. Otherwise, if the scheme OPT contains some bad pairs of calls, we create a scheme S by copying OPT and applying the following changes (depicted in Figure 3.6).

(1) Select an arbitrary bad pair of calls $((u, v), (u, w))$, where (u, v) was placed in time unit t_1, and (u, w) was placed in time unit $t_2 > t_1$.

(2) Swap the order of the calls in the pair: use the edge (u, w) in time unit t_1 and the edge (u, v) in time unit t_2.

(3) Leave the rest of the calls unchanged (even if it causes idling).

This transformation is visualized in Figure 3.6, where subfigure (a) shows the order of calls in scheme OPT and subfigure (b) shows the order of calls in scheme S. After the modifications, the scheme S will finish broadcasting in the same amount of time, because by Lemma 3.3.1, the vertex v does not place any calls after being informed. Moreover, the above transformation will reduce the number of bad pairs of calls in scheme S by at least 1. Hence, after a finite number of steps, scheme S will have optimal broadcast time containing no bad pairs of calls, thus, proving the lemma. □

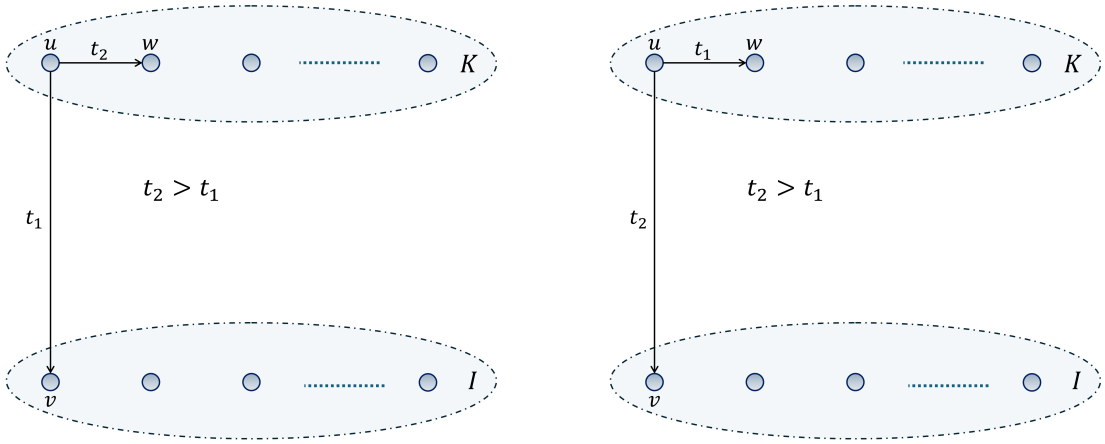


Figure 3.6: Example of a transformation discussed in Lemma 3.3.2.

Let M be a proper t-star-matching induced by an optimal broadcast scheme S satisfying the conditions of Lemma 3.3.2 (as well as Lemma 3.3.1), and let N_i and $d_i, 1 \leq i \leq n$ be the set and the number of vertices covered by $v_i \in K$ in M, respectively. Also, let $b(S)$ denote the broadcast time achieved by scheme S. The following corollary follows from Lemma 3.3.2.

Corollary 3.3.3. *In the optimal broadcast scheme S, a vertex $v_i \in K$, starts placing calls to vertices in N_i in time unit $b(S) - d_i + 1$. Additionally, the calls can be placed in an arbitrary order without affecting the broadcast time.*

Now we will analyze the broadcast time $b(S)$ achieved by broadcast scheme S. It is obvious that if no calls are placed from a vertex in the clique to the independent set then, theoretically, by the time the broadcasting is over $2^{b(S)}$ vertices could be informed in the clique (informed vertices

are doubled every time unit). Now assume that a vertex $u \in K$ spends the last p time units of the broadcasting process placing calls to some vertices on the independent set I. Since, theoretically, these p time units could be used to inform $2^p - 1$ vertices in the clique, then the overall number of vertices that will be informed in the clique would be at most $2^{b(S)} - 2^p + 1$. Thus, according to Corollary 3.3.3, the following number is an upper bound on the number of informed vertices in the clique after $b(S)$ time units.

$$2^{b(S)} - 2^{d_1} + 1 - 2^{d_2} + 1 - ... - 2^{d_n} + 1 = 2^{b(S)} - \sum_{i=1}^{n} 2^{d_i} + n \tag{7}$$

As the scheme S is a valid broadcast scheme and informs all the vertices in the graph within $b(S)$ time units, the following inequality is obvious.

$$2^{b(S)} - \sum_{i=1}^{n} 2^{d_i} + n \geq n \tag{8}$$

$$2^{b(S)} - \sum_{i=1}^{n} 2^{d_i} \geq 0 \tag{9}$$

$$2^{b(S)} \geq \sum_{i=1}^{n} 2^{d_i} \tag{10}$$

Definition 3.3.4. *For a given proper t-star-matching M, let the **cost** of M, denoted by $C(M)$, be $C(M) = \sum_{i=1}^{n} 2^{d_i}$, where d_i is the degree of vertex $v_i \in K$ in the subgraph induced by M.*

Let M^* be a proper t-star-matching that minimizes the cost $C(M^*)$. We define the following decision problem for the star-matching with minimum cost.

Problem 3.3.5. *Minimum cost star-matching*

Instance: *(G, c), where $G = (V, E)$ is a split graph, and c is a natural number.*

Output: *"Yes" if there exists a proper star-matching M^* of G such that $C(M^*) \leq c$; "No" otherwise.*

The below corollary follows from the Equation (10).

Corollary 3.3.6. *The broadcast time of a split graph G from an originator $v \in K$ is lower bounded by $b(v, G) \geq \lceil \log(C(M^*)) \rceil$. Moreover, there exists an optimal broadcast scheme S, such that the*

edges between K and I in the broadcast tree of S exactly induce M^*.

Hence, the problem of finding the minimum broadcast time of a given split graph can be solved if the minimum cost star-matching problem is solved. Based on everything discussed previously, the following exact algorithm for the minimum broadcast time problem could be derived.

Algorithm 5 Exact Algorithm

Input A split graph G, a clique K, an independent set I, an originator $v \in K$, and a natural number t

Output "Yes" if $b(v, G) \leq t$; "No" otherwise.

```
1:  procedure SplitGraphBroadcast
2:      Find a proper t-star-matching M with minimum cost C(M)
3:      if 2^t ≤ C(M) then
4:          return "No"
5:      end if

6:      d ← []
7:      d[i] ← degree of v_i ∈ K in M
8:      Sort the degree array d in a non-increasing fashion
9:      Broadcast in the clique prioritizing the vertices based on their order in d

10:     if a vertex v_i ∈ K is not informed in time unit t − d_i + 1 then
11:         return "No"
12:     else
13:         Use the edges of v_i in M in arbitrary order starting from time unit t − d_i + 1
14:     end if
15:     return "Yes"
16: end procedure
```

Algorithm 5 follows the claims made previously in this section. The only exception is the conditional operation (if) on line 10. The statement checks if all the vertices that need to start placing calls towards the independent set are informed before their determined time unit. If this is not the case then the broadcasting cannot terminate within the given amount of time.

To sum up, Algorithm 5 gives a strategy that can generate an exact algorithm if the proper star-matching with minimum cost can be found. However, for any selected matching strategy the algorithm will produce a heuristic, the performance of which will only depend on the cost of the selected matching. For instance, if we used the same proper star-matching with min maxdegree that was used in the approximation algorithm introduced earlier in Sections 3.2.2 and 3.2.3, it would generate a valid heuristic. Clearly, it is possible to have many idle vertices in the approximation

algorithm that we discussed earlier, depending on the structure of the graph instance. Thus, the heuristic will generate broadcast times not worse than those of the 2-approximation algorithm, as it will have the same behavior without any vertex idling. However, we were unable to provide a better approximation ratio for that algorithm. We will discuss the heuristic in detail further in this chapter.

3.3.1 Split graphs with known broadcast times

As mentioned in the previous section, Algorithm 5 will return the minimum broadcast time for split graphs if the proper star-matching with minimum cost can be found. In this section we will consider two subclasses of split graphs for which the proper star-matching with minimum cost is known, hence, resulting in exact polynomial-time algorithms.

First, we consider split graphs where the degree of each vertex in the independent set is one. In this case, each vertex in the independent set should be covered by the only adjacent vertex in the clique. Hence, there exists only one proper star-matching for these split graphs.

Let G be a split graph with a clique partition K and an independent set partition I, such that $deg(u) = 1$ for every $u \in I$, $|K| = n$, $|I| = m$. Assume K contains vertices $v_1, v_2, \ldots, v_n$. We will consider two cases: the originator in K and I.

(1) Let some $v_j \in K$ be the originator. As argued above, there exists only one proper star-matching M of G. Moreover, we know that each vertex $v_i \in K$ has degree d_i in the subgraph induced by M, where $d_i = deg(v_i) - n + 1$ is the number of vertices in the independent set adjacent to v_i. Clearly, by the Corollary 3.3.6, the broadcast time of graph G from the originator v is equal to $b(v_j, G) = \lceil \log(\sum_{i=1}^{n} 2^{d_i}) \rceil$.

(2) Let $w \in I$ be the originator. Since every vertex in the independent set has degree one, in the first time unit the vertex w will pass the message to its only neighbor, say $v_j \in K$, and will be idle after that. Let G' be a split graph resulting after removing the vertex w from G. Obviously, $b(w, G) = 1 + b(v_j, G')$. Whereas, by the previous case, $b(v_j, G') = \lceil \log(\sum_{i=1}^{n} 2^{d'_i}) \rceil$, where d'_i is the number of vertices in the independent set of G' adjacent to $v_i \in K$. Hence, $b(w, G) = 1 + \lceil \log(\sum_{i=1}^{n} 2^{d'_i}) \rceil$.

Now, we consider split graphs where each vertex in the clique has at most one adjacent vertex

in the independent set. In this case, there can exist several proper star-matchings. However, in the subgraph induced by any proper star-matching, each vertex in the clique either has a degree 0 or 1. Moreover, there exist exactly m vertices in the clique with degree 1, where $m = |I|$.

Let G be a split graph with a clique partition K and an independent set partition I, such that $|K| = n$, $|I| = m$, and $deg(u) \leq n$ for every $u \in K$. Let $v \in K$ be the originator. Let d_i be the number of vertices in the independent set adjacent to $v_i \in K$. From the above-mentioned argument, the following equation is implied.

$$\sum_{i=1}^{n} 2^{d_i} = m \cdot 2^1 + (n - m) \cdot 2^0 = n + m \tag{11}$$

Hence, by the Corollary 3.3.6, the broadcast time of graph G from the originator v is equal to $b(v, G) = \lceil \log(n + m) \rceil$.

3.3.2 Split graphs with achievable lower bound of the broadcast time

In this section, we will discuss an infinite subfamily of split graphs that have broadcast time equal to the lower bound. As we mentioned earlier in this study, the broadcast time is lower bounded by $\lceil \log n \rceil$, where $n = |V|$. So for a split graph G, with clique partition $|K| = n$ and independent set partition $|I| = m$, the broadcast time is lower bounded by $\lceil \log(n + m) \rceil$.

Assume that for a given graph G and an originator $v \in K$, $b(v, G) = \lceil \log(n+m) \rceil$. Let M be a proper t-star-matching induced by an optimal broadcast scheme for G, and let N_i and $d_i, 1 \leq i \leq n$ be the set and the number of vertices covered by $v_i \in K$ in M, respectively. For any vertex v_i, $d(v_i)$ denotes the number of vertices in I adjacent to v_i, or more formally, $d(v_i) = |N(v_i) \cap I|$, where $N(v_i)$ is the open neighborhood of v_i. Note that $0 \leq d_i \leq d(v_i)$ for any i. The following is straightforward from the Equation (10).

$$2^{\lceil \log(n+m) \rceil} \geq \sum_{i=1}^{n} 2^{d_i} \tag{12}$$

Since $d_i \leq d(v_i)$, then we can claim that $b(v, G) = \lceil \log(n+m) \rceil$ if

$$2^{\lceil \log(n+m) \rceil} \geq \sum_{i=1}^{n} 2^{d(v_i)} \tag{13}$$

Thus, any split graph satisfying the above-mentioned inequality will have a broadcast time equal to the lower bound of $\lceil \log(n+m) \rceil$. For instance, let $n = 2^k$, $m = 2^k + 1$, for a positive integer k, and let each vertex in the clique have at most 2 neighbors in the independent set ($d(v_i) \leq 2$, for any $1 \leq i \leq n$). In that case, we can see that $2^{\lceil \log(n+m) \rceil} = 2^{k+2}$ and $\sum_{i=1}^{n} 2^{d(v_i)} \leq 4 \cdot n = 2^{k+2}$. Hence, Equation (13) holds for these graphs, and we can claim that they have broadcast time of $b(v, G) = \lceil \log(n+m) \rceil$.

3.4 Broadcasting heuristic in split graphs

The heuristic, which we will introduce shortly, is based on the analysis conducted in the previous section. Given a connected split graph G with a clique partition K, an independent set partition I, and an arbitrary originator $v \in V(G)$ we can consider the following two cases.

(1) The originator is from the clique: $v \in K$.

(2) The originator is from the independent set: $v \in I$.

However, for both of these cases, the algorithm strategy is the same. The only difference is that for an originator in the independent set, the originator will not be considered in the step of finding a star-matching.

3.4.1 Algorithm description

The main goal of the algorithm is to avoid vertex idling. Moreover, the broadcast scheme also prioritizes informing vertices in the clique and will start to broadcast to the independent set as late as possible. The vertices in the clique will be given some priority based on their degree in the proper star-matching. Vertices that have more neighbors to inform in the independent set will be informed earlier. However, instead of directly finding a broadcast time for the given split graph, we will try

to find an answer to the broadcast time decision problem. Given a graph $G = (V, E)$, the originator $v \in V$, and a natural number b, the broadcast time decision problem returns "Yes" if $b(v, G) \leq b$ and "No" otherwise. The pseudocode for the algorithm that follows the analysis in Section 3.3 is presented in Algorithm 6.

Algorithm 6 Heuristic for Broadcasting in Split Graphs

Input A split graph G, a clique K, an independent set I and an originator $v \in V(G)$
Output A broadcast time b.

procedure FINDBROADCASTTIME
  $G' \leftarrow G$
  if $v \in I$ **then**
    Remove v from G'
  end if
  Find a star-matching M of G' (Algorithm 2)
  $d \leftarrow []$
  $d[i] \leftarrow$ degree of $v_i \in K$ in M
  Sort the degree array d in a non-increasing order
  Do a binary search on range $[\log |V|, |V| - 1]$ by invoking BROADCAST procedure
end procedure
procedure BROADCAST($d[]$, b) ▷ **Input**: sorted degree array d, natural number b
  ▷ **Output** "Yes" if $b(v, G) \leq b$; "No" otherwise
  $q \leftarrow v$ ▷ Set of informed vertices
  $i \leftarrow 1$
  while $i \leq b$ **do** ▷ Iterate through rounds
    Remove $d[0, |q| - 1]$ from d ▷ Inform first $|q|$ vertices in the list d
    Add $d[0, |q| - 1]$ to q
    if a vertex $v_i \in K$ with $d_i = t - i + 1$ is not informed **then**
      return "No" ▷ Vertex v_i fails to start broadcasting to I
    else
      Remove v_i from q
    end if
  end while

  if d is not empty **then**
    return "No" ▷ Not all vertices of K are informed
  end if
  return "Yes"
end procedure

The time complexity of the FINDBROADCASTTIME procedure in Algorithm 6 consists of several components. First, as we already discussed in Section 3.2.1, finding the star-matching will have a complexity of $\mathcal{O}(|E||V| \log |I|)$. Next, sorting of the degree array will have a complexity

of $\mathcal{O}(|K|\log|K|)$. Any operation in the sub-procedure BROADCAST can be implemented with constant time complexity. Thus, overall, the procedure BROADCAST will have complexity $\mathcal{O}(b)$ caused by the loop on line 16. Hence, overall complexity of the heuristic will be $\mathcal{O}(|E||V|\log|I|)+$ $\mathcal{O}(|K|\log|K|)+\mathcal{O}(b\log|V|)=\mathcal{O}(|E||V|\log|I|)$. Lastly, the Algorithm 6 will be invoked during a binary search on range $[\log|V|,|V|-1]$.

As per Section 3.3, the fact that Algorithm 6 uses a star-matching with minimum maxdegree makes it a polynomial-time 2-approximation algorithm for broadcasting. Since Algorithm 6 avoids idle vertices, it returns equally good or better broadcast times than the algorithms in Section 3.2. Unfortunately, we were unable to provide a better approximation ratio for the algorithm. The algorithm will be empirically analyzed in Section 3.6.

3.5 Broadcasting in (k,l)-graphs

To understand if the strategies discussed in Sections 3.2 and 3.4 can be applied to other graph families, we are going to consider a natural generalization of split graphs called (k,l)-graphs (Hell, Klein, Protti, & Tito, 2001). In a (k,l)-graph (Brandstädt, 1996), the vertex set can be partitioned into k independent sets and l cliques. Such partitioning is referred to as (k,l)-partition. Note that split graphs are exactly $(1,1)$-graphs. Currently, there exist efficient recognition algorithms for split graphs (Golumbic, 2004), $(2,1)$-graphs and $(1,2)$-graphs (Brandstädt & Szymczak, 1998). Moreover, for $k\geq 3$ or $l\geq 3$, recognizing (k,l)-graphs is shown to be NP-complete (Brandstädt, 1996, 1998). Note that the class of $(k,0)$-graphs is precisely the class of k-colorable graphs.

Split graphs are particularly interesting for research because they are chordal, which is not true for all (k,l)-graphs. Hence, chordal (k,l)-graphs are an interesting case of (k,l)-graphs. Hell, Klein, Nogueira, and Protti (2004) particularly discussed the chordal (k,l)-graphs. They showed a forbidden subgraph characterization of chordal (k,l)-graphs, as well as introduced a polynomial-time recognition algorithm.

Further, we will introduce a heuristic for the broadcast time problem in (k,l)-graphs. Note that every connected graph $G=(V,E)$ is a $(0,|V|-1)$-graph. Hence, even though we designed our algorithm to target (k,l)-graphs with small k and l, it will still work for arbitrary graphs.

Consider a connected graph $G = (V, E)$ such that the vertex set can be partitioned into k independent sets $I_1, I_2, ..., I_k$ and l cliques $K_1, K_2, ..., K_l$. Let $\mathcal{I} = I_1 \cup I_2 \cup ... \cup I_k$ be the set of all vertices in the independent sets, and let $\mathcal{K} = K_1 \cup K_2 \cup ... \cup K_l$ be the set of all clique vertices. We occasionally may denote a (k, l)-graphs as $G = (\mathcal{I}, \mathcal{K}, E)$. Before proceeding to the actual algorithm, we will introduce several important tools that will be used as keystones.

3.5.1 Finding a tree-matching

Recall that a *proper t-star-matching* of a split graph G is a perfect t-star-matching such that any vertex $v \in K$ is a center of a star. We are going to use a natural generalization of a proper t-star-matching for (k, l)-graphs.

Definition 3.5.1.1. *A **tree-matching** of a (k, l)-graph $G = (\mathcal{I}, \mathcal{K}, E)$ is a collection of mutually vertex-disjoint subtrees of G (a forest), such that the root of each tree is from $\mathcal{K}$, every vertex of each tree other than the root is from $\mathcal{I}$, and every vertex of G belongs to exactly one tree.*

Note that a tree-matching of a split graph is a proper star-matching.

The goal is to find a tree-matching that can be used to induce an efficient broadcast scheme. For that, we are going to introduce a mechanism that tries to ensure that vertices in $\mathcal{K}$ have mutually balanced trees. We designed two recursive procedures to construct the tree-matching level by level.

Given a simple graph $G = (V, E)$, a *matching* M is a set of pairwise non-adjacent edges in G, i.e. none of the edges share a common vertex. A *bipartite matching* $M(U, W)$, on sets of vertices U and W, is a matching where for every edge $(u, w) \in M(U, W)$, $u \in U$ and $w \in W$. In other words, $M(U, W)$ is a matching of the bipartite graph induced by the partitions U and W. If two partitions are clear from the context we may sometimes refer to it as just M. A *maximal matching* of G is a matching that is not a subset of another matching in G. For two sets of vertices, U and W, $MM(U, W)$ denotes a *maximal bipartite matching*. To find the next level of each tree we are going to select multiple maximal bipartite matchings as long as such exist.

The procedure takes as an input a graph $G = (V, E)$, a dominant set $U \subset V$, and a secondary set $W \subset V$, such that $U \cap W = \emptyset$. It keeps searching for maximal matchings between sets U and W until every vertex in W is covered or is impossible to cover. The collection of all selected maximal

matchings, referred to as a *recursive maximal matching* from U to W, is denoted by $RMM(U, W)$ (note that the notation takes an ordered pair). The procedure RECURSIVEMAXIMALMATCHING is described in Algorithm 7. Figure 3.7 visualizes the procedure RECURSIVEMAXIMALMATCHING.

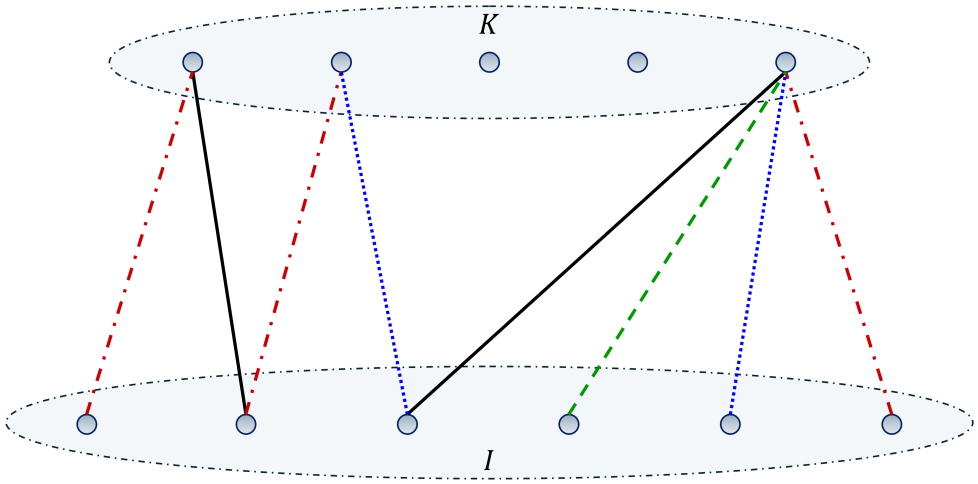


Figure 3.7: Example of a recursive maximal matching $RMM(K, I)$. Red, blue, and green edges are those selected in iterations one, two, and three, respectively.

The main algorithm for finding a tree-matching starts by finding a recursive maximal matching R from $\mathcal{K}$ to $\mathcal{I}$. It is clear that each vertex in $\mathcal{I}$ that is not covered by R is not adjacent to any vertex in $\mathcal{K}$. Hence, the vertices that are still not covered, participate in the next round of finding a recursive maximal matching. This time, they will be covered by the vertices in $\mathcal{I}$ that were previously selected into R. If we continue this process, eventually, every vertex will be covered and all used recursive matchings together will induce a tree-matching. The algorithm TREEMATCHING is formally presented in Algorithm 8. An example of a tree-matching returned by the Algorithm 8 is presented in Figure 3.8.

3.5.2 Broadcasting heursitic

Given a (k, l)-graph $G = (V, E)$ with partitions $\mathcal{I}$ and $\mathcal{K}$, let M be a tree-matching of G. Below we introduce several variables that will be used to design the algorithm. The order of calls will be based on the values of these variables. For a clique K, current time unit t, and an attempted broadcast time b:

- $X(K)$ denotes the set of informed vertices in K.

Algorithm 7 Finding a recursive maximal matching

Input Graph $G = (V, E)$, two sets $U, W \subset V$, $U \cap W = \emptyset$
Output $RMM(U, W)$

procedure RECUSRIVEMAXIMALMATCHING
  $R \leftarrow \{\}$
  while W is not empty **do**
    Find a maximal matching $S = MM(U, W)$
    if $S = \emptyset$ **then**
      return R
    end if
    $R \leftarrow R \cup S$
    for each $(u, w) \in S$, where $u \in U, w \in W$ **do**
      Remove w from W
    end for
  end while
  return R
end procedure

- The *estimated error* of K, denoted by $e(K)$, is the number of time units after b that K is estimated to be fully informed, assuming that a call is placed to a vertex in K in time unit t.

$$e(K) = \max\left\{0, t + 1 + \log\left(\frac{|K|}{1 + |X(K)|}\right) - b\right\}$$

- The *secondary error* of K, denoted by $e_2(K)$, is the number of time units after b that K is estimated to be fully informed, assuming that a call is placed to a vertex in K in time unit $t + 1$.

$$e_2(K) = \max\left\{0, t + 2 + \log\left(\frac{|K|}{1 + |X(K)|}\right) - b\right\}$$

For a vertex $v \in V$, current time unit t, and an attempted broadcast time b:

- $T(v)$ denotes the tree in M that v belongs to.
- $N_k(v)$ is the set of cliques that v has a neighbor in (excluding the clique that v belongs to).
- $UC(v)$ denotes the set of children of v in $T(v)$ that are currently not informed.
- $w(v)$ denotes the *current weight* of v in each time unit of the broadcasting process such that:
  - if v is a leaf, $w(v) = 0$,

Algorithm 8 Finding a tree-matching

Input (k, l)-graph $G = (\mathcal{I}, \mathcal{K}, E)$
Output A tree-matching M

procedure TREEMATCHING
  $M \leftarrow \{\}$
  $U \leftarrow \mathcal{K}$
  $W \leftarrow \mathcal{I}$
  while W is not empty **do**
    Find $R = RMM(U, W)$
    $M \leftarrow M \cup R$
    $U \leftarrow R \cap W$
    $W \leftarrow W \setminus R$
  end while
  return M
end procedure

- otherwise, $w(v) = \max\{i + w(u_i) | u_i \in UC(v)\}$, where $UC(v)$ is sorted in non-increasing order of children's current weights.

- The *deadline* of v, denoted by $f(v)$, is the largest number of time units v can wait before starting to inform the vertices in $UC(v)$. Clearly,

$$f(v) = b - w(v) - t + 1$$

and v needs to inform vertices in $UC(v)$ when $f(v) = 0$ as latest.

For a vertex v in some clique K_i, $1 \leq i \leq l$, the *cost* of v, denoted by $c(v)$, is the aggregation between the deadline of v, the estimated error of K_i, and the secondary error of all neighboring cliques.

$$\begin{aligned} c(v) &= \frac{f(v)}{1 + e(K_i) + \frac{\max\{e_2(K) | K \in N_k(v)\}}{2}} \\ &= \frac{2f(v)}{2 + 2e(K_i) + \max\{e_2(K) | K \in N_k(v)\}} \end{aligned} \tag{14}$$

It is easy to see that vertices with smaller deadlines should be informed earlier to finish broadcasting in their uninformed subtrees on time. Similarly, cliques with higher errors are more likely to need calls from outside the clique itself to finish the broadcast process on time. Hence, the cost of a

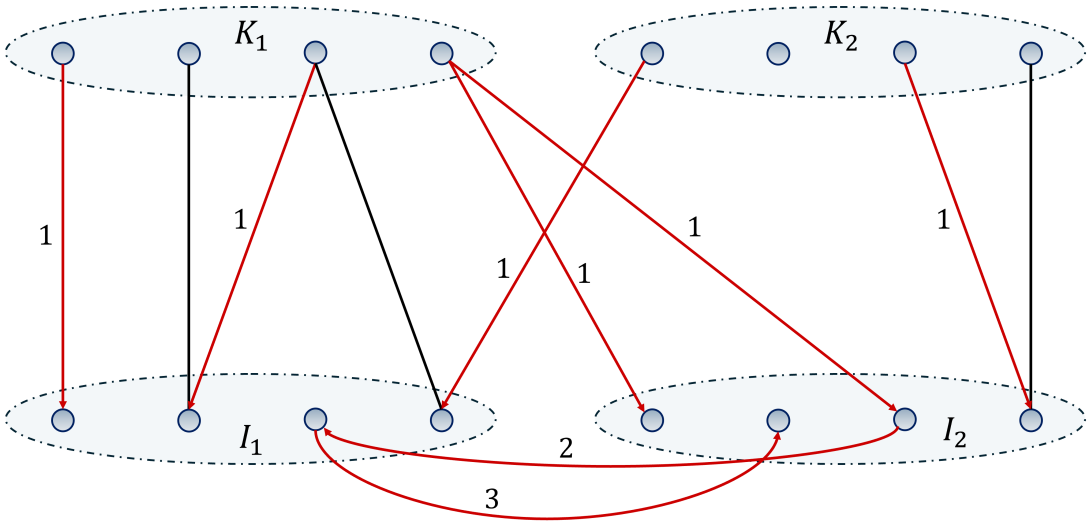


Figure 3.8: Example of a tree-matching on a $(2, 2)$-graph. Red edges form the tree-matching, and the labels represent the round when the edge was selected.

vertex $v \in \mathcal{K}$ defines the priority of v in comparison to other vertices in $\mathcal{K}$, i.e., vertices with less cost should be informed first. Whenever an informed vertex does not have an uninformed neighbor in a clique, it places a call towards a neighbor in $\mathcal{I}$ with the smallest deadline. Overall, this greedy strategy can be summarized with the following points.

In each time unit $t \leq b$, an informed vertex u follows the greedy broadcast scheme below:

(1) If $f(u) > 0$:

   (a) u informs its neighbor in $\mathcal{K}$ with the smallest cost. If two vertices in different cliques have the same cost, select the one with a higher estimated error. If two vertices are selected from the same clique, pick arbitrarily.

   (b) If no such vertex is found, then u informs its neighbor in $\mathcal{I}$ with the smallest deadline (with ties broken arbitrarily).

(2) If $f(u) = 0$:

   (a) u informs its child in $UC(u)$ with the smallest deadline.

The broadcasting algorithm for (k, l)-graphs is formally presented in Algorithm 9.

The time complexity of the Algorithm 9 consists of several components. First, finding the recursive maximal matching will have a complexity of $\mathcal{O}(|E||V||\mathcal{I}|)$. Next, the main part of the

Algorithm 9 Broadcasting (k, l)-graphs

Input (k, l)-graph $G = (V, E)$, an originator $v \in V$, and a positive integer b
Output "Yes" if $b(v, G) \leq b$; "No" otherwise

procedure BROADCAST
  Find a tree-matching M of G (Algorithm 8)
  $t \leftarrow 1$
  $Z \leftarrow \{v\}$ ▷ The set of informed vertices
  $X \leftarrow V \setminus \{v\}$ ▷ The set of uninformed vertices
  while $t \leq b$ **do**
    for each $u \in Z$ **do**
      $A \leftarrow \mathcal{K} \cap N(u) \cap X$
      $B \leftarrow \mathcal{I} \cap N(u) \cap X$
      if $f(u) = 0$ **then**
        Select $s \in UC(u)$ with the min $f(s)$
      else if $A \neq \emptyset$ **then**
        Select $s \in A$ with the min $c(s)$ and max $e(K)$
      else if $B \neq \emptyset$ **then**
        Select $s \in B$ with the min $f(s)$
      end if
      $X \leftarrow X \setminus s$
      $Z \leftarrow Z \cup \{s\}$
    end for
    $t \leftarrow t + 1$
    Update the deadlines and the costs
  end while
  if X is not empty **then**
    return "No" ▷ Not all vertices of K are informed
  end if
  return "Yes"
end procedure

algorithm has $\mathcal{O}(|V|b)$ iterations. Whereas, each operation inside the loop itself, if carefully implemented, can be executed in constant time. Hence, overall complexity of the heuristic will be $\mathcal{O}(|E||V||\mathcal{I}|) + \mathcal{O}(b|V|) = \mathcal{O}(|E||V||\mathcal{I}|)$.

3.6 Experiments and results

In this section, we will discuss the results achieved after different experiments with the proposed heuristics, as well as discuss some special cases of (k, l)-graphs.

3.6.1 Fully Connected Trees

A *fully connected tree (FCT)* (Gholami et al., 2023) is a graph $G = (V, E)$ which can be partitioned into n rooted trees $T_1, T_2, ..., T_n$, such that the roots of the trees form a clique K_n.

Lemma 3.6.1.1. *Any fully connected tree is a* $(2, 1)$*-graph.*

Proof. It is easy to see that the vertex set of any tree T can be partitioned into 2 independent sets by assigning the vertices on odd levels to the first set, and the vertices on even levels to the other set. Moreover, since there are no edges between two trees in a fully connected tree (other than edges between the roots), all of the trees can be partitioned into the same two independent sets. Hence, by assigning the roots to a clique and the rest of the vertices to two independent sets, we can see that any fully connected tree is a $(2, 1)$-graph. □

Lemma 3.6.1.2. *Algorithm 9 is a polynomial-time exact algorithm for broadcasting when applied to fully connected trees.*

Proof. Note that a fully connected tree does not contain a cycle going through a vertex in an independent set, and hence, has a unique tree-matching. Moreover, since there is only one clique, the vertices in the clique have the same estimated error factor. Thus, the cost of these vertices is only defined by the deadline which is based on the broadcast time of the attached tree. Thus, we can see that the proposed algorithm for (k, l)-graphs, when applied to fully connected trees, induces the known polynomial-time exact algorithm (*GFCT* by Gholami et al. (2023)). □

3.6.2 Split graphs

In the first group of experiments, we test the results of the algorithm devised in Section 3.4 on random graphs with a somewhat balanced predefined structure. The goal of these experiments is to validate the baseline against what we are going to compare the (k, l)-graph heuristic. The main purpose of the second group of experiments is to validate the motivation of the heuristics by comparing the results of the algorithms devised in Sections 3.4 and 3.5. Lastly, we study the behavior of the proposed heuristics on graph instances that were generated following some predefined structure. We refer to the (k, l)-graph heuristic as **RMM** and to the split graphs heuristic as **SMH**.

Bounded degree split graphs

First, we will present experimental results on split graphs where vertices in the independent set have bounded degrees. We call such graphs d*-restricted split graphs*, where d is the maximum degree out of all vertices in the independent set. This subfamily of graphs represents networks where nodes have a limit on the number of connections to the network core. To simulate such networks, we create split graph instances with a given number of vertices (N), a predefined number of vertices in the clique (n), and independent set (m) partitions. After the partition is defined, vertices in the independent set are connected by edges to a randomly selected set of vertices in the clique, such that the size of the set is limited by a given input value d (degree bound). Later, we run *SMH* heuristic on these graph instances with random originators. The algorithm is executed on up to 50 generated graph instances for each set of input parameters, and for each execution, the resulting broadcast time is retrieved. We report the minimum (*SMH: Min*), the maximum (*SMH: Max*), and the average (*SMH: Avg*) of all resulting broadcast times for each input parameter set. Tables 3.1 and 3.2 present experimental results on 5-restricted and 15-restricted split graphs, respectively.

We can see that in some cases the average broadcast time is equal to the lower bound of the broadcasting, which means that the algorithm found the minimum broadcast time in all executions. Moreover, from the tables, it is clear that the average, minimum, and maximum broadcast times grow as the size of the clique decreases while keeping the number of vertices constant. In the majority of the cases when the clique and the independent set have almost equal cardinalities, the algorithm correctly found a broadcast scheme with the optimal broadcast time, which could mean that as the clique size decreases the optimal broadcast time of the graph is not equal to the lower bound anymore. This is explained by the fact that a smaller clique size means a higher maximum degree of star-matchings. Overall, we can conclude that the algorithm *SMH* performed very well for k-restricted split graphs.

Unstructured graphs

For the second round of experiments, we create random split graphs with different densities. Firstly, the vertex set of the graph is randomly divided into two partitions: the clique and the independent set. After that, edges are added between every pair of clique vertices. Lastly, with a given probability p (density parameter), an edge is added between every pair of vertices, where one vertex is from the independent set and the other vertex is from the clique.

We run the algorithms for different graph sizes and three values of the density parameter: 0.2 (sparse graphs), 0.5 (dense graphs), and 0.9 (near-complete graphs). The broadcasting is initiated from a randomly selected originator. The results for different values of the density parameter p and different numbers of vertices N are presented in Table 3.3. The algorithm is executed on up to 50 generated graph instances for each pair of input parameters (N, p), and for each execution, the resulting broadcast time is retrieved. In Table 3.3, we present the maximums (*RMM: Max* and *SMH: Max*), and the averages (*RMM: Avg* and *SMH: Avg*) of all resulting broadcast times.

We can see that *RMM* heuristic returns broadcast times that are rather close to the lower bound of $\lceil \log N \rceil$. Moreover, we observe that *RMM* heuristic returns better average broadcast times than the *SMH* heuristic in all cases. In some cases, *RMM* even results in better maximum broadcast time than *SMH*. This means, that for randomly created split graphs or when the structure of the split graph is unknown, the proposed heuristic *RMM* is preferable to *SMH*. Given that we do not know the exact value of the minimum broadcast time, the results achieved by heuristic *RMM* are considered successful.

Threshold graphs

In the case of random split graphs, vertices in the independent set are connected to their neighbors with uniformly random probability and share the same set of limitations. Since these graphs represent fairly balanced networks, it is still interesting to study graphs with imbalanced adjacency between vertices.

Our next group of experiments was performed on threshold graphs (Hammer, 1977). Threshold graphs are constructed from a single vertex graph by addition of vertices. Each added vertex is

either not connected to any of the existing vertices (*isolated*) or is connected to all of the existing vertices (*dominating*).

Claim 3.6.2.1. *All threshold graphs split.*

Proof. Let every vertex that was selected as isolated during the construction process be in the set I and every vertex that was selected as dominating be in the set K. From the construction of graph G we know that no pair of vertices in I is connected by an edge, hence, making it an independent set. Now, let $u, v \in K$ be an arbitrary pair of vertices in K, such that, without loss of generality (wlog), v was added later than u. Since v was selected to be dominating, then it would be connected to every vertex that was already in the graph, including u. Hence, K will be a clique in G. Thus, we showed that for any threshold graph G, there exists a decomposition into a clique and an independent set, making it a split graph. □

For the sake of our experiments, we construct threshold graphs, where each added vertex is randomly selected to be isolated or dominating with a uniform probability. However, the last added vertex is always a dominating vertex, because otherwise the graph would not be connected. Notice that this process will generate sparse graphs with an approximately equal number of vertices in two partitions. Moreover, the process generates graphs with a wide distribution of vertex degrees, where vertices in the independent set have more neighbors if added earlier, and vice versa for the vertices in the clique. As an unbalanced subfamily of split graphs, threshold graphs are one of the worst possible input graphs for our heuristic. Hence, we study the behavior of our algorithm on these graphs to understand the worst-case scenario.

The results achieved by the heuristics from arbitrary originators are provided in Table 3.4. For a given graph with N vertices, we analyze the broadcast time achieved by heuristic *RMM* by comparing it with the lower bounds of the minimum broadcast time and the broadcast time of algorithm *SMH*. As we can see, in most of the cases *RMM* returns slightly worse broadcast times than those of *SMH*. This means that in the case of split graphs which have a very unbalanced degree distribution of the vertices in the clique, algorithm *SMH* is preferable to algorithm *RMM*.

Table 3.1: Experimental results for 5-restricted split graphs of different number of vertices (N), different cardinalities of the clique (n) and the independent set (m).

N	n	m	$\lceil \log N \rceil$	SMH: Min	SMH: Max	SMH: Avg
260	150	110	9	9	9	9.00
260	130	130	9	9	9	9.00
260	110	150	9	9	9	9.00
260	90	170	9	9	10	9.58
260	70	190	9	10	11	10.14
260	50	210	9	11	12	11.10
260	30	230	9	14	16	14.20
260	20	240	9	17	19	18.18
260	15	245	9	22	25	22.98
130	85	45	8	8	8	8.00
130	75	55	8	8	8	8.00
130	65	65	8	8	9	8.04
130	55	75	8	8	9	8.02
130	45	85	8	8	10	8.30
130	35	95	8	9	10	9.04
130	25	105	8	10	11	10.06
130	15	115	8	13	14	13.06
130	10	120	8	16	19	16.96
70	45	25	7	7	7	7.00
70	40	30	7	7	7	7.00
70	35	35	7	7	8	7.06
70	30	40	7	7	7	7.00
70	25	45	7	7	8	7.10
70	20	50	7	7	8	7.96
70	15	55	7	8	9	8.54
70	10	60	7	10	12	10.34
70	5	65	7	16	20	17.04
40	30	10	6	6	6	6.00
40	25	15	6	6	6	6.00
40	20	20	6	6	7	6.16
40	15	25	6	6	8	6.10
40	10	30	6	7	8	7.04
40	7	33	6	8	10	8.38
40	5	35	6	10	11	10.18

Table 3.2: Experimental results for 15-restricted split graphs of different number of vertices (N), different cardinalities of the clique (n) and the independent set (m).

N	n	m	$\lceil \log N \rceil$	SMH: Min	SMH: Max	SMH: Avg
1100	600	500	11	11	11	11.00
1100	500	600	11	11	11	11.00
1100	400	700	11	11	12	11.40
1100	300	800	11	12	12	12.00
1100	225	875	11	12	13	12.70
1100	150	950	11	15	15	15.00
1100	100	1000	11	18	19	18.06
1100	50	1050	11	28	31	29.70
550	300	250	10	10	10	10.00
550	250	300	10	10	10	10.00
550	220	330	10	10	10	10.00
550	190	360	10	10	11	10.00
550	180	370	10	10	11	10.45
550	150	400	10	11	11	11.00
550	100	450	10	12	12	12.00
550	75	475	10	14	14	14.00
550	50	500	10	16	18	16.80
550	25	525	10	27	32	27.62
260	170	90	9	9	9	9.00
260	150	110	9	9	9	9.00
260	130	130	9	9	9	9.00
260	110	150	9	9	9	9.00
260	90	170	9	9	10	9.04
260	70	190	9	9	10	9.74
260	50	210	9	11	11	11.00
260	30	230	9	13	14	13.26
130	75	55	8	8	8	8.00
130	65	65	8	8	8	8.00
130	55	75	8	8	8	8.00
130	45	85	8	8	8	8.00
130	30	100	8	9	9	9.00
130	22	108	8	10	10	10.00
130	15	115	8	12	13	12.32

Table 3.3: Experimental results for random split graphs of different number of vertices (N) and density of edges (p).

N	p	$\lceil \log N \rceil$	RMM: Max	SMH: Max	RMM: Avg	SMH: Avg
1000	0.2	10	11	11	**10.30**	10.50
500	0.2	9	10	10	**9.30**	9.60
100	0.2	7	8	8	**7.05**	7.20
80	0.2	7	**7**	8	**7.00**	7.20
50	0.2	6	**7**	8	**6.20**	6.50
32	0.2	5	**7**	8	**5.55**	6.00
1000	0.5	10	11	11	**10.20**	10.80
500	0.5	9	10	10	**9.30**	9.60
100	0.5	7	**7**	8	**7.00**	7.30
80	0.5	7	**7**	8	**7.00**	7.03
50	0.5	6	7	7	**6.05**	6.38
32	0.5	5	6	6	**5.35**	5.50
1000	0.9	10	11	11	**10.20**	10.70
500	0.9	9	10	10	**9.20**	9.30
100	0.9	7	**7**	8	**7.00**	7.40
80	0.9	7	**7**	8	**7.00**	7.03
50	0.9	6	7	7	**6.10**	6.43
32	0.9	5	6	6	**5.45**	5.50

Table 3.4: Experimental results for randomly generated threshold graphs on different number of vertices N.

N	$\lceil \log N \rceil$	RMM	SMH
1280	11	21	13
1088	11	15	16
1040	11	18	13
1028	11	19	14
768	10	15	11
576	10	19	13
528	10	16	13
516	10	17	12
320	9	19	13

N	$\lceil \log N \rceil$	RMM	SMH
260	9	17	13
192	8	17	11
144	8	14	10
132	8	9	8
80	7	8	7
68	7	8	8
48	6	11	7
36	6	9	7
20	5	6	6

Chapter 4

Recursively Decomposable Graphs

This chapter presents our contributions to broadcasting in networks with some recursive structures. We will introduce an exact polynomial-time algorithm on closed chains of rings. We will also propose a new perspective to devising optimal broadcast schemes. Lastly, we will study how this approach applies to different recursive structures.

4.1 Introduction

In literature, several compound structures of graphs were addressed in the context of both minimum broadcast time and minimum broadcast graph problems.

Farley (1979) has constructed broadcast graphs recursively by combining two or three smaller broadcast graphs and showed $B(n) \leq \frac{n}{2}\lceil \log n \rceil$. Chau and Liestman (1985) generalized this construction by using up to seven small broadcast graphs. In (Khachatrian & Harutounian, 1990), a compounding method has been introduced for constructing mbgs using vertex covers of graphs. This method constructs new broadcast graphs by forming the compound of several known broadcast graphs. Further, Bermond, Fraigniaud, and Peters (1995) generalized the compounding method to a unified way of constructing broadcast graphs for any n.

The minimum broadcast time problem was also addressed on some compound structures like fully connected trees (Gholami, Harutyunyan, & Maraachlian, 2023), tree of cycles (Harutyunyan & Maraachlian, 2009b), tree of cliques (Maraachlian, 2010), etc.

If a graph G can be recursively defined as a compound of graphs $G_1, G_2, \ldots, G_p$, for some positive integer p, then we say that G is *recursively decomposable*. Note that this concept is different from the idea of recursive graph families defined by Biggs, Damerell, and Sands (1972).

Definition 4.1.1. *A cactus is a connected graph in which any two cycles have at most one vertex in common. Equivalently, each edge in a cactus graph is on at most one cycle.*

Cactus graphs are an interesting class of graphs, as reported in Čevnik and Žerovnik (2017), they are a common generalization of trees and ring networks (Ben-Moshe, Dvir, Segal, & Tamir, 2012; Elenbogen & Fink, 2007). Moreover, cacti are recursively decomposable. If we consider any cycle on a cactus, all subgraphs attached to that cycle are also cacti. The minimum broadcast time problem is widely believed to be NP-hard on general cacti, but to the best of our knowledge, there is no proof for that in the literature. Moreover, the minimum broadcast time problem is believed to be NP-hard when restricted to some subfamilies of cactus graphs, e.g. flower (k-cycle) graphs. Bhabak (2014) studied broadcasting on k-cycle graph where k cycles of arbitrary lengths are connected by a central vertex. The author designed a constant approximation algorithm to find the broadcast time of an arbitrary k-cycle graph. For some simpler cactus graphs there exist polynomial-time exact broadcasting algorithms, e.g. unicyclic graphs (Harutyunyan & Maraachlian, 2007, 2008), tree of cycles (Harutyunyan & Maraachlian, 2009b), necklace graphs; to be defined later (Harutyunyan, Laza, & Maraachlian, 2009), and k-restricted cacti[1] (Čevnik & Žerovnik, 2017).

A *necklace graph* is a collection of cycles, where each consecutive pair of cycles is connected by one vertex. In (Harutyunyan et al., 2009), the authors devised an algorithm to solve the broadcasting problem on an arbitrary necklace graph. Assuming that the originator is on cycle C_i and the end cycles are cycles C_1 and C_k, the authors split the necklace graphs into two necklace graphs: one being the subgraph starting at cycle C_1 and ending at cycle C_i, and the other one being the subgraph starting at C_i and ending at C_k. After solving the problem on these two graphs, the solutions are put together to construct the solution for the original graph. Another interesting case arises when the two end cycles of the necklace graph get connected at a vertex. We call such graphs *closed chains of rings* or *closed necklace graphs*.

[1] A *k-restricted cactus* (or simply *k-cactus*) graph is a cactus graph where each vertex belongs to at most k cycles.

The algorithm in (Harutyunyan et al., 2009) is one of the few broadcasting algorithms in the literature that is devised recursively. However, those algorithms are designed for simpler recursive structures and are hard to adapt to other recursively decomposable graphs. The main contributions of this chapter are an exact polynomial-time broadcasting algorithm on closed chains of rings and a systematic approach to developing recursive broadcasting algorithms. For the sake of completeness of this manuscript, Section 4.2 includes some auxiliary results by Harutyunyan, Laza, and Maraachlian (2009) to study the limitations of the introduced recursive broadcasting strategy.

4.2 Cycle and an attached graph

Consider a graph G made up of a cycle C_N and a graph G', such that both graphs have a vertex v_c in common. The vertex v_c is called the connecting vertex.

Theorem 4.2.1. *For any originator v_0 on C_N, there exists an optimal broadcast scheme that first sends the information along the shorter path towards vertex v_c and then along the longer path.*

Proof. Assume there does not exist any broadcast scheme that first sends the information along the shorter path towards v_c. Let S be an optimal broadcast scheme. We construct a broadcast scheme S' which sends the information along the shorter path in the first time unit. Clearly, using broadcast scheme S', v_c will be informed at least a time unit earlier than in S. Assume that in S', v_c is informed in time unit t. In time unit $t+1$, v_c can send the information to its uninformed neighbor (if both neighbors are informed, then both paths from v_0 to v_c are equal) and start broadcasting in G' after time unit $t+2$. This means that all vertices on C_N will be informed after the minimum amount of time ($\lceil \frac{N}{2} \rceil$), whereas G' will be informed at least as early as in S. Hence, S' is an optimal broadcast scheme. □

4.2.1 The broadcast time of a cycle with a tree

In this section, the goal is to calculate the broadcast time of the graph composed of a cycle with a tree T attached to one of its vertices. In other words, the root of the tree T is a vertex on the cycle. The analysis is limited to the case where the root of the tree has a degree of 2. Consider a graph G made up of a cycle C_N and a tree T connected to a vertex v_m (see Figure 4.1). Let v_0

be the originator. According to Theorem 4.2.1, an optimum broadcast scheme can be obtained by first informing the vertex on the shortest path from v_0 to v_m. After that, the broadcast scheme is straightforward until vertex v_m is informed, as every informed vertex informs its only uninformed neighbor on the cycle. The only case when a vertex has more than one choice is when v_m gets informed. Let v'_m be the other vertex that was informed at the same time as v_m, and P be the number of uninformed vertices on the cycle C_N at the time v_m was informed. Assume that the tree T has a broadcast time of t, and let T_1 and T_2 be the two subtrees connected to v_m with broadcast times t_1 and t_2 respectively.

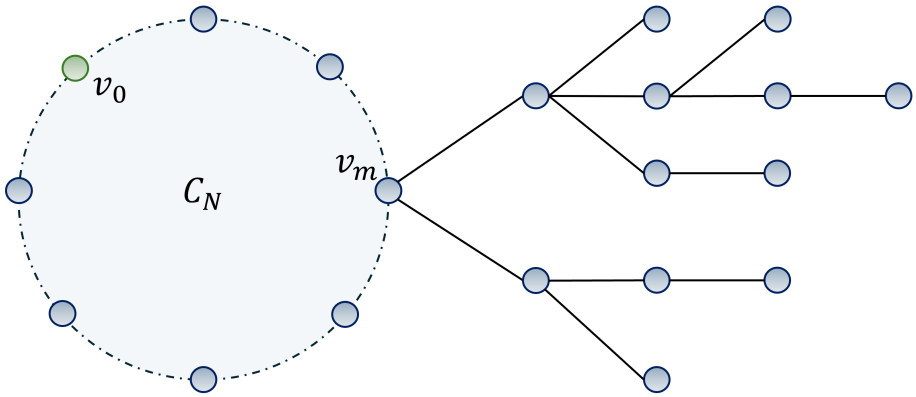


Figure 4.1: Example of a cycle C_N and a tree attached to vertex v_m.

Obviously, the broadcast time of G depends on the values of t, t_1, t_2 and P.

(1) $t < \lceil \frac{P}{2} \rceil$: An optimum broadcast scheme is obtained when v_m first informs its neighbor on C_N and then starts broadcasting in the tree, resulting in broadcast time equal to that of C_N.

(2) $t > \lceil \frac{P}{2} \rceil$: An optimum broadcast scheme is obtained when in the first two time units v_m informs its neighbors in the tree and then informs the neighbor on C_N, hence, $b(G) = d(v_0, v_m) + t$.

(3) $t = \lceil \frac{P}{2} \rceil$: This case requires careful observation of the structure of the tree. We are going to discuss the following possible cases.

   (a) P is even, $t = P/2$, $t_1 = t - 1$, $t_2 \leq t - 2$.

   (b) P is even, $t = P/2$, $t_1 = t - 2$, $t_2 = t - 2$.

(c) P is odd, $t = (P+1)/2$, $t_1 = t-1$, $t_2 = t-2$.

(d) P is odd, $t = (P+1)/2$, $t_1 = t-2$, $t_2 = t-2$.

For all these cases, informing the vertex on the cycle first, then T_1, and lastly T_2 will result in $b(G) = d(v_0, v_m) + t + 1$. The same broadcast time can be achieved when the vertex on the cycle is informed after T_1 and T_2. However, the broadcast time of two subtrees in the broadcast tree is important to consider. In the first case, the difference is 2 time units, in the second case it is 1.

(e) P is odd, $t = (P+1)/2$, $t_1 = t-2$, $t_2 \leq t-3$.

In this case, the optimum broadcast scheme informs the root of T_1, then the vertex on the cycle, and finally the root of the tree T_2. The broadcast time of this scheme is $b(G) = d(v_0, v_m) + t$. The broadcast tree will have two subtrees, both having broadcast time of $d(v_0, v_m) + t - 2$.

The following observations can be made based on the previous results.

Assume that we have a graph G made up of a cycle C and a graph G', such that G' is attached to C at v_c and the degree of v_c in G' is equal to 2. Assume that T is a broadcast tree of G' for the originator v_c. Let r_1 and r_2 be the two children of v_c and T_1 and T_2 be the two subtrees of T rooted at r_1 and r_2 correspondingly.

Lemma 4.2.1.1. *For an arbitrary given originator v_0 on C, other than v_c, the broadcast time $b(v_0, G) = b(v_0, G_T)$, where graph G_T is made up of the cycle C and the broadcast tree T attached at v_c. If there are several broadcast trees, then the tree that has $b(v_1, T_1) > b(v_2, T_2) + 2$ must be chosen.*

Note that all the broadcast trees of a necklace graph have roots of degree 2, since they are on a cycle where all vertices have degree 2.

4.3 Broadcasting in a closed chain of rings

A closed chain of rings is a chain of at least three rings (necklace) where the two end cycles of the chain have a common vertex (Fig. 4.2). All vertices that belong to two cycles are said to be

connecting, whereas vertices that are on exactly one cycle are *non-connecting* or *internal*. In this section, we will use the results achieved in previous sections to introduce an exact polynomial-time algorithm for the broadcast time problem in a closed chain of rings. This makes closed chains of rings the first non-k-cacti graph family with intersecting cycles, for which an exact broadcasting algorithm was devised.

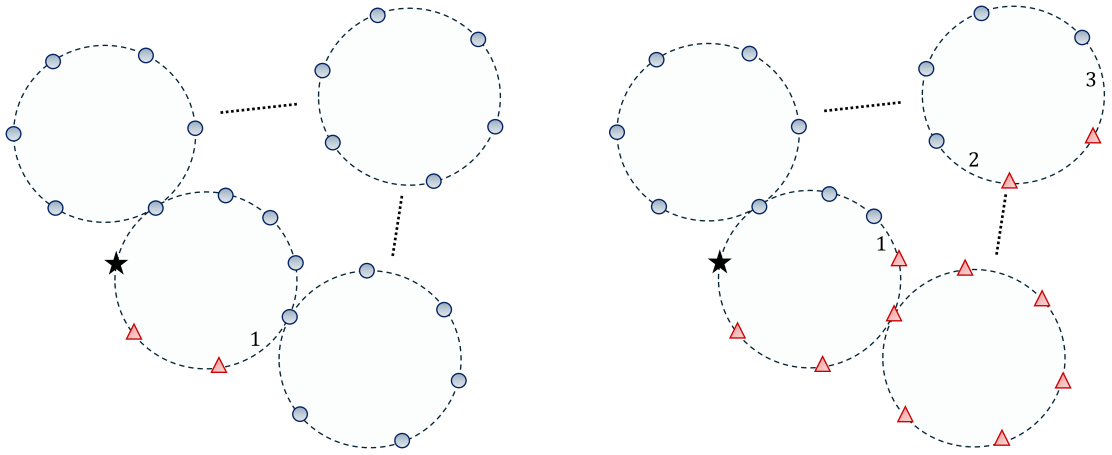
4.3.1 Broadcasting from an internal vertex

Let G be a closed chain of rings, v be a non-connecting vertex in G, and T be a broadcast tree that results in a minimum broadcast time in the graph G from the originator v. Graph G consists of n cycles $C_1, C_2, ..., C_n$. Assume, wlog, that $v \in C_1$. Let T_1 and T_2 be two subtrees of the tree T rooted at two neighbors of v (v_1 and v_2, respectively).

It is easy to see that there exist either one or three vertices in T_1, such that they have a neighbor in T_2. The case when there is only one such vertex is only possible if all vertices of T_1 are on C_1, and none of the connecting vertices of C_1 are in T_1. The case when there are three such vertices is possible if T_1 contains one of the connecting vertices on C_1. We will consider each case separately.

(1) There exists one vertex $u \in T_1$ that has a neighbor w in T_2. In this case, splitting the cycle C_1 by cutting the edge (u, w) will result in a graph G', which will have the same broadcast time as graph G. In G', two disjoint subgraphs G'_1 and G'_2 correspond to the trees T_1 and T_2 and are connected to the originator v. Graph G'_1 is a path graph, whereas, graph G'_2 is a chain of rings with a tree attached to the end cycle. Clearly, G' is a 2-restricted cactus, and we can find the broadcast time in linear time using the algorithm in (Čevnik & Žerovnik, 2017).

(2) There exist three vertices $u_1, u_2, u_3 \in T_1$ that have neighbors w_1, w_2, w_3 in T_2. This case is only possible when two of the vertices are in the same cycle C_i, $i \neq 1$, and the other vertex is on cycle C_1. Let u_1 and u_2 be in the same cycle C_i and u_3 be on C_1. Note that u_1 and u_2 may be adjacent to the same vertex in T_2, i.e. $w_1 = w_2$. In this case, splitting the cycle C_i by cutting the edges (u_1, w_1) and (u_2, w_2), and splitting the cycle C_1 by cutting the edge (u_3, w_3) will result in a graph G' which will have the same broadcast time as graph G. Similarly, G' is a 2-restricted cactus.



(a) (b)

Figure 4.2: Examples of subtrees of the broadcast three rooted at an internal originator (black star vertex). The vertices of each subtree are marked with different colors and shapes. Numbered edges are the edges that need to be cut.

Hence, given the vertex or the triplet of vertices that define an optimal broadcast tree, we can construct the broadcast scheme in a bottom-up approach similar to broadcasting in the open chain of rings. The set of possible split vertices is limited and can be considered exhaustively.

(1) Recall that in the first case, the vertex u was on the cycle C_1. Hence, only $|C_1|$ vertices can be considered.

(2) Whereas, in the second case, both vertices u_1 and u_2 belong to the same cycle C_i and $u_3 \in C_1$. Thus, there exist $|C_1| \cdot \sum_{1<i\leq n} \binom{|C_i|}{2}$ possible pairs of vertices.

Fig. 4.2 shows examples of each possible case of two subtrees T_1 and T_2. Note that, the examples do not represent an actual broadcast scheme and aim to visualize the possible cuts.

4.3.2 Broadcasting from a connecting vertex

Let G be a closed chain of rings, v be a connecting vertex in G, and T be a broadcast tree that results in a minimum broadcast time in the graph G from the originator v. Graph G consists of n cycles $C_1, C_2, ..., C_n$. Assume, wlog, that v is the connecting vertex between cycles C_1 and C_2. Let T_1, T_2, T_3 and T_4 be four subtrees of the tree T rooted at four neighbors v_1, v_2, v_3, v_4 of v, such that $v_1, v_2 \in C_1$ and $v_3, v_4 \in C_2$.

Similar to the mechanism used in Section 4.3.1, we are going to find the possible number of vertices that define the broadcast tree.

Other than v, the other connecting vertex on cycle C_1 (as well as the one on C_2) can belong to only one of the subtrees, making the other subtree in that cycle a simple path. Assume, wlog, that T_1 and T_3 are simple paths from v to u_1 and u_3, respectively. Both u_1 and u_3 have adjacent vertices (w_1 and w_3) that can be either in T_2 or T_4. Whereas, the split between T_2 and T_4 can be defined analogously to the way described in Section 4.3.1: there can exist either one or two vertices that have a neighbor in the other subtree. We will consider each case separately.

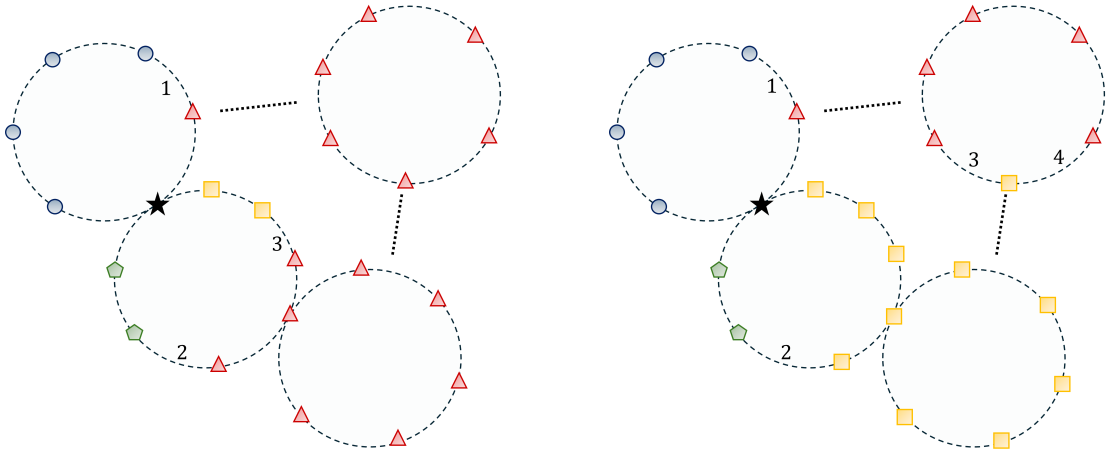
(1) There exists one vertex $u \in T_2$ that has a neighbor w in T_4. This case is only possible if $u \in C_1$ or if $w \in C_2$, are not a connecting vertex. In this case, cutting the edges $(u, w), (u_1, w_1), (u_3, w_3)$ will result in a graph G' which will have the same broadcast time as graph G. In G', four disjoint subgraphs G'_1, G'_2, G'_3 and G'_4 correspond to the trees T_1, T_2, T_3 and T_4 and are connected to the originator v. Again, graph G' is a 2-restricted cactus graph.

(2) There exist two vertices $u_2, u_4 \in T_2$ that have neighbors w_2, w_4 in T_4. Moreover, this case is only possible when both u_2 and u_4 are in the same cycle C_i, $i > 2$. Recall that u_2 and u_4 may be adjacent to the same vertex in T_4, i.e. $w_2 = w_4$. In this case, cutting the edges $(u_1, w_1), (u_2, w_2), (u_3, w_3)$ and (u_4, w_4) will result in a 2-restricted cactus graph G' which will have the same broadcast time as graph G.

Hence, given 3 or 4 vertices that define an optimal broadcast tree, we can construct the broadcast scheme in a bottom-up approach similar to broadcasting in the open chain of rings. The set of possible split vertices is limited and can be considered exhaustively. Vertices u_1 and u_3 have $|C_1|$ and $|C_3|$ possible cases respectively.

(1) Recall that in the first case, the vertex u was on cycle C_1 or C_2. Hence, only $|C_1| + |C_2|$ vertices can be considered.

(2) Whereas, in the second case, both vertices u_2 and u_4 belong to the same cycle C_i. Thus, there exist $\sum_{2<i\leq n} \binom{|C_i|}{2}$ possible pairs of vertices.



(a) (b)

Figure 4.3: Examples of subtrees of the broadcast three rooted at an internal originator (black star vertex). The vertices of each subtree are marked with different colors and shapes. Numbered edges are the edges that need to be cut.

Hence, it is possible to construct an optimal broadcast scheme for broadcasting in closed chains of rings in polynomial time. Fig. 4.3 shows examples similar to the previous section.

4.4 Recursive broadcasting

In general, dividing a graph into smaller subgraphs, solving the broadcast problem in the smaller graphs, and combining the solutions does not always yield an optimum result. However, this was possible in necklace graphs provided that some care is taken while choosing the broadcast tree in case there is more than one option. The foundation of the strategy that was applied to necklace graphs was the fact that the connecting vertex has degree 2 in the attached graph. Whereas, given a recursively decomposable graph where the connecting vertices have higher degrees, the problem becomes more complicated. With some knowledge about the compounding (recursive) step, it is possible to come up with an optimal broadcast scheme. The keystone of our approach is focusing on a specific optimal broadcast scheme, since there may be many optimal schemes solving the subproblems that do not generate an optimal solution for the original problem.

4.4.1 Lazy broadcasting approach

Recall that an informed vertex v is idle in time unit t of a broadcasting process if v does not make a call in time t.

Definition 4.4.1.1. *A broadcast scheme is called **busy** (or **non-lazy**), if any informed vertex sends a message to one of its uninformed neighbors, providing there are any, during each round. On the contrary, a broadcast scheme is called **lazy**, if there exists a vertex v, which remained idle even though it had an uninformed neighbor.*

Busy schemes guarantee that as long as there remains an uninformed neighbor, vertices are never idle. It can very easily be seen that for every lazy broadcast scheme, there exists a busy broadcast scheme that guarantees at most the same broadcast time. All one needs to do is shift the calls to cover all the idle time units of informed vertices. That is why, most of the time, any study related to broadcasting is working with busy schemes, and lazy schemes are barely considered. However, we will further show that lazy schemes can be very useful for designing optimal broadcast schemes, especially when relying on a recursive approach.

Let S be a broadcast scheme for a graph $G = (V, E)$, where $v \in V$ is the originator. Also, let $b(S)$ denote the broadcast time achieved by S. Given a vertex $u \in V$, let $t_u(S)$ denote the time unit in S when u is informed. For the originator, $t_v(S) = 0$.

Definition 4.4.1.2. *The **degree of freedom** of u in S (denoted by $\alpha(u, S)$) is the smallest integer $i \geq 0$, such that u is idle in time unit $t_u(S) + i + 1$. The degree of freedom of S is the degree of freedom of the originator; formally $\alpha(S) = \alpha(v, S)$.*

Less formally, the degree of freedom is the number of time units that pass after being informed when a vertex is first idle. Note that, if $\alpha(S) = 0$, then S can be transformed to a scheme S' by removing the delay in the first time unit, such that $b(S') = b(S) - 1$ and both induce the same broadcast tree. Thus, for our purposes, we will assume that $\alpha(S) = \alpha(v, S) \geq 1$ (only for the originator, not other vertices).

Remark 4.4.1.3. *The degree of freedom of a vertex u in a busy broadcast scheme is equal to the number of vertices that were informed by u.*

Definition 4.4.1.4. *A broadcast scheme S* ***respects freedom****, if for any broadcast scheme S', such that $b(S) = b(S')$, then $\alpha(S) \leq \alpha(S')$. In other words, S has the smallest possible degree of freedom while maintaining the same broadcast time.*

4.4.2 Bridge-connected graphs

Let G be a graph comprising an edge (v_1, v_2) and two vertex-disjoint subgraphs G_1 and G_2 with the following properties:

- $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$.
- For an arbitrary vertex $v \in G_1$ and an arbitrary vertex $u \in G_2$, $(v, u) \notin E(G)$.

In other words, (v_1, v_2) is a cut edge[1] of G, and G_1, G_2 are the components that are created if (v_1, v_2) is removed.

Fig. 4.4 shows an example of the above-described graph. We may denote such a graph as $G = (v_1, v_2, G_1, G_2)$.

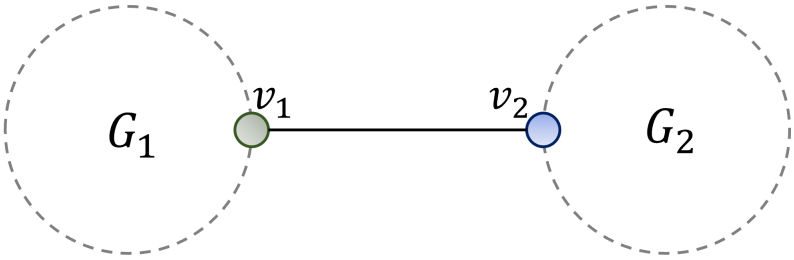


Figure 4.4: Visualization of bridge-connected graphs $G = (v_1, v_2, G_1, G_2)$.

Consider a broadcasting process in G that originates from v_1. The following bounds on the broadcast time are easy to see:

$$\max\{b(v_1, G_1), 1 + b(v_2, G_2)\} \leq b(v_1, G) \leq 1 + \max\{b(v_1, G_1), b(v_2, G_2)\}$$

The lower bound emerges when we combine the lower bounds on the broadcast times of two subgraphs:

[1] An edge $e \in G$ is called a *cut edge* (or a *bridge*) if the removal of e from G increases the number of connected components of G.

- Since G_1 is a subgraph of G, $b(v_1, G) \geq b(v_1, G_1)$.
- Consider the graph G' comprising G_2 and the edge (v_1, v_2), without the rest of graph G_1. Obviously, $b(v_1, G) \geq b(v_1, G') = 1 + b(v_1, G_2)$.

Whereas, the upper bound results from the following straightforward broadcast scheme:

- In the first time unit, v_1 informs v_2.
- Then both v_1 and v_2 broadcast in their corresponding subgraphs.

However, when $b(v_1, G_1) \leq b(v_2, G_2)$, the upper and lower bounds are equal, hence, $b(v_1, G) = 1 + b(v_2, G_2)$.

We are more interested in resolving the case when $b(v_1, G_1) > b(v_2, G_2)$. More specifically, our goal is to answer if $b(v_1, G) = b(v_1, G_1)$ or not.

Lemma 4.4.2.1. *Let $S_1 = \mathcal{S}(v_1, G_1)$ and $S_2 = \mathcal{S}(v_2, G_2)$, such that S_1 respects freedom. If $b(S_1) \geq 1 + \alpha(S_1) + b(S_2)$, then $b(v_1, G) = b(S_1)$. Otherwise, $b(v_1, G) = 1 + b(S_1)$.*

Proof. Consider the following broadcast scheme X.

(1) If $b(S_1) < 1 + \alpha(S_1) + b(S_2)$, then

- In the first time unit, v_1 informs v_2.
- After that, v_1 and v_2 follow S_1 and S_2, respectively.

(2) If $b(S_1) \geq 1 + \alpha(S_1) + b(S_2)$, then

- First $\alpha(S_1)$ time units, v_1 follows S_1.
- In time unit $1 + \alpha(S_1)$, v_1 informs v_2.
- After that, v_1 and v_2 follow S_1 and S_2, respectively.

If case 1 is followed, then the broadcast time achieved by X will be $b(X) = 1 + b(S_1)$.

In case 2, v_2 is informed in time unit $1 + \alpha(S_1)$. Hence, following S_2, G_2 will be fully informed at time unit $1 + \alpha(S_1) + b(S_2)$. By the execution condition of case 2, $1 + \alpha(S_1) + b(S_2) \leq b(S_1)$, hence, G_2 will be fully informed by time unit $b(S_1)$. When it comes to G_1, we can see that v_1

follows S_1 at all times, but time unit $1 + \alpha(S_1)$. However, by the definition of $\alpha(S_1)$, v_1 is idle in time unit $1 + \alpha(S_1)$. Hence, the fact that v_1 informs v_2 will not affect the broadcast time of G_1, which will be fully informed by time unit $b(S_1)$. Since both components of G are fully informed, $b(X) = b(S_1)$.

We will prove the lemma by contradiction.

Let $Y = \mathcal{S}(v_1, G)$ be an optimal broadcast scheme of G. If $b(Y) = 1 + b(S_1)$, then $b(X) \leq b(Y)$, making X an optimal broadcast scheme too.

Assume $b(Y) = b(S_1)$ and $b(X) = 1 + b(S_1)$. Recall that $t_{v_2}(Y)$ denotes the time unit when v_2 is informed in Y. Since $b(Y) = b(S_1)$, we can claim that:

$$\begin{aligned} t_{v_2}(Y) + b(S_2) &\leq b(Y) = b(S_1) \\ t_{v_2}(Y) &\leq b(S_1) - b(S_2) \end{aligned} \tag{15}$$

Moreover, since $b(X) = 1 + b(S_1)$, then the first case of scheme X took place. Thus:

$$\begin{aligned} b(S_1) &< 1 + \alpha(S_1) + b(S_2) \\ \alpha(S_1) &> b(S_1) - b(S_2) + 1 \\ \alpha(S_1) &> t_{v_2}(Y) + 1 \end{aligned} \tag{16}$$

Let us gather all the calls of Y made within the subgraph G_1 into a new scheme S_1' while conserving laziness. Clearly, $b(S_1') \leq b(Y) = b(S_1)$, making S_1' and optimal broadcast scheme for G_1 originating from v_1. Moreover, since in Y, v_1 informed v_2 in time unit $t_{v_2}(Y)$, in the same time unit v_1 will be idle in S'. Hence, by Equation (16):

$$\alpha(S_1') \leq t_{v_2}(Y) - 1 < \alpha(S_1) \tag{17}$$

This makes S_1' an optimal broadcast scheme with a degree of freedom smaller than that of S_1. Which is a contradiction. □

4.4.3 Path-connected graphs

Consider a graph G consisting of a path P with vertices $v_1, v_2, ..., v_l$ (in the path order), and l pairwise vertex-disjoint graphs $G_1, G_2, ..., G_l$ attached to P with the following properties:

- $v_i \in V(G_i)$, for any $1 \leq i \leq l$.
- For any $1 \leq i \leq l$, $1 \leq j \leq l$, $i \neq j$, an arbitrary vertex $v \in G_i$ and an arbitrary vertex $u \in G_j$, $(v, u) \notin E(G)$.

Fig. 4.5 gives an example of the above-described graph. We may denote such a graph as $G = (P, G_1, G_2, ..., G_l)$.

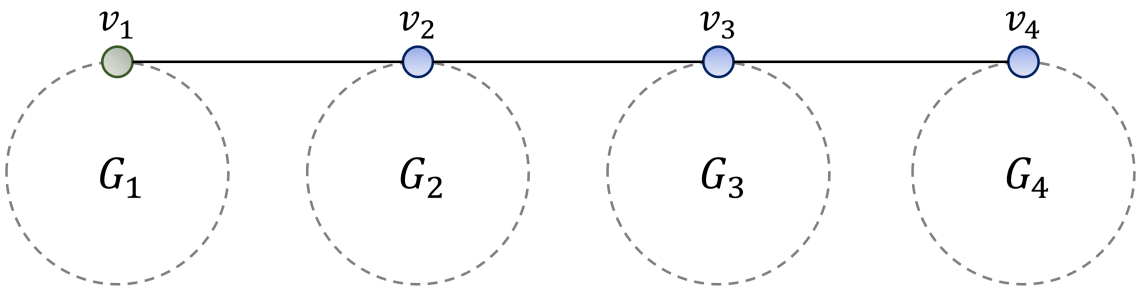


Figure 4.5: Visualization of path-connected graphs $G = (P, G_1, G_2, G_3, G_4)$, where v_1, v_2, v_3, v_4 form the path P.

Given a graph $G = (P, G_1, G_2, ..., G_l)$, where $P = \{v_1, v_2, ..., v_l\}$, and an originator v_1, we are going to consider two separate problems.

(1) **Fixed-deadline message dissemination problem:** For a fixed positive integer t, the goal is to find the maximum integer $1 \leq j \leq l$ for which $G_1, G_2, ..., G_j$ can be fully informed within t time units. We denote j by $f(v_1, G, t)$, or simply $f(G, t)$, if v_1 is clear from the context.

(2) **Multicasting problem:** For an integer $1 \leq j \leq l$, consider the minimum multicast time problem where $U_j = V(G_1) \cup V(G_2) \cup ... \cup V(G_j)$ is the set of terminals. The goal is to find the minimum multicast time, denoted by $b(v_1, U_j, G)$.

Fixed-deadline message dissemination problem

Given a graph $G = (P, G_1, G_2, ..., G_l)$, where $P = \{v_1, v_2, ..., v_l\}$, let $S_1, S_2, ..., S_l$ be optimal broadcast schemes respecting freedom for graphs $G_1, G_2, ..., G_l$ originating from $v_1, v_2, ..., v_l$, respectively. Algorithm 10 finds the maximum integer $1 \leq j \leq l$, for which a fixed integer deadline t is sufficient.

Algorithm 10 Fixed-deadline message dissemination in Path-connected graphs

Input

1. A graph $G = (P, G_1, G_2, ..., G_l)$, where $P = \{v_1, v_2, ..., v_l\}$.
2. $S_1, S_2, ..., S_l$, where S_i is an optimal broadcast scheme respecting freedom of G_i originating from v_i.
3. A deadline t.

Output Positive integer $1 \leq j \leq l$.

```
1:  procedure FIXEDDEADLINEPATH(G, t)
2:      t_{v_i} is the time unit when v_i is informed
3:      i ← 1
4:      while 1 + t_{v_i} + b(S_i) ≤ t do
5:          if i == l then
6:              return l
7:          end if
8:          if 2 + t_{v_i} + b(S_i) ≤ t then
9:              v_i informs v_{i+1}, as soon as it gets informed
10:             t_{v_{i+1}} = t_{v_i} + 1
11:         else
12:             v_i informs v_{i+1}, α(S_i) time units after being informed
13:             t_{v_{i+1}} = t_{v_i} + 1 + α(S_i)
14:         end if
15:         v_i follows S_i in all other time units
16:         i ← i + 1
17:     end while
18:     return i − 1
19: end procedure
```

Lemma 4.4.3.1. *Let k be the value returned by procedure* FIXEDDEADLINEPATH. *Integer k is an optimal solution for the fixed-deadline message dissemination problem, or simply $f(v_1, G, t) = k$.*

Proof. We are going to prove this by contradiction. Let X be the scheme described by FIXEDDEADLINEPATH. Assume a scheme Y exists that, within t time unit, informs all subgraphs up to G_m, such as $k < m \leq l$. Let $2 \leq j \leq k + 1$ be the smallest integer for which $t_{v_j}(X) > t_{v_j}(Y)$.

If such j does not exist, then v_{k+1} in X was informed in the same time unit or earlier than in Y. However, since the loop on Step 4 of Algorithm 10 was not executed, it means that $1 + t_{v_{k+1}}(X) + b(S_i) > t$. Which implies that

$$1 + t_{v_{k+1}}(Y) + b(S_i) \geq 1 + t_{v_{k+1}}(X) + b(S_i) > t \tag{18}$$

Moreover, $1 + t_{v_{k+1}}(Y) + b(S_i)$ represents the theoretical minimum time required for G_{k+1} to be fully informed, rendering Y invalid.

If such j does exist, then v_{j-1} was informed at most in the same time unit in X than in Y; $t_{v_{j-1}}(X) \leq t_{v_{j-1}}(Y)$. There are two options on when v_j is informed in X:

- v_j was informed instantly after v_{j-1} was informed (Step 9 of Algorithm 10). In this case, $t_{v_j}(X) = 1 + t_{v_{j-1}}(X) \leq 1 + t_{v_{j-1}}(Y) \leq t_{v_j}(Y)$. Thus, this case is not possible.

- v_j was informed $1 + \alpha(S_{j-1})$ after v_{j-1} was informed (Step 12 of Algorithm 10). First of all, this case implies that G_{j-1} should have been informed within $b(S_{j-1}) = b(v_{j-1}, G_{j-1})$ time units after v_{j-1} was informed, both in X and Y. Additionally,

$$t_{v_j}(X) = 1 + \alpha(S_{j-1}) + t_{v_{j-1}}(X) \leq 1 + \alpha(S_{j-1}) + t_{v_{j-1}}(Y) \tag{19}$$

  Let β be the number of time units after receiving the message that v_{j-1} spent placing calls to its neighbors in G_{j-1} following scheme Y.

$$t_{v_j}(Y) = 1 + \beta + t_{v_{j-1}}(Y) \geq 1 + \beta + t_{v_{j-1}}(X) \tag{20}$$

  Since by assumption $t_{v_j}(Y) < t_{v_j}(X)$, Equations (19) and (20) imply that $\beta < \alpha(S_{j-1})$. Let S' be the broadcast scheme for G_{j-1}, originating from v_{j-1}, induced by Y (preserving idle time units). As mentioned, $b(S') = b(v_{j-1}, G_{j-1})$, making scheme S' an optimal broadcast scheme for G_{j-1} originating from v_{j-1}. Since v_{j-1} informed v_j after β time units, it may remain idle in that time unit in S'. Therefore, $\alpha(S') = \beta < \alpha(S_{j-1})$. Which contradicts the definition of S.

□

Multicasting problem

Again, for a graph $G = (P, G_1, G_2, ..., G_l)$, where $P = \{v_1, v_2, ..., v_l\}$, let $S_1, S_2, ..., S_l$ be any optimal broadcast schemes respecting freedom for graphs $G_1, G_2, ..., G_l$ originating from $v_1, v_2, ..., v_l$, respectively. Given an integer $1 \leq j \leq l$, the goal is to find the minimum multicast time $b(v_1, U_j, G)$, where $U_j = V(G_1) \cup V(G_2) \cup ... \cup V(G_j)$ is the set of terminals.

Lemma 4.4.3.2. $\max\{i - 1 + b(S_i) | 1 \leq i \leq j\} \leq b(v_1, U_j, G) \leq \max\{i + b(S_i) | 1 \leq i \leq j\}$

Proof. First, we prove the lower bound. Let $k - 1 + b(S_k) = \max\{i - 1 + b(S_i) | 1 \leq i \leq j\}$, where $1 \leq k \leq j$. The fastest way v_k can be informed is via a direct path from v_1, requiring $k - 1$ time units. After that, to fully inform G_k, k requires at least $b(S_k)$ time units. Since $V(G_k) \in U_j$, we can claim that:

$$b(v_1, U_j, G) \geq k - 1 + b(S_k) = \max\{i - 1 + b(S_i) | 1 \leq i \leq j\} \tag{21}$$

To prove the upper bound, consider the following simple multicast scheme X:

(1) Each vertex v_i, $1 \leq i < j$, informs its neighbor on path P, namely v_{i+1}, in the next time unit after being informed.

(2) In all other cases, v_i, $1 \leq i \leq j$, follows S_i.

In X, vertex v_i, for all $1 \leq i < j$, is informed via a direct path from the originator v_1 and receives the message in time unit $i - 1$. Then, it spends a time unit informing its neighbor on the path P, to start broadcasting in G_i after time unit i. Broadcasting in G_i will be over by time unit $i + b(S_i)$. Hence:

$$b(v_1, U_j, G) \leq b(X) = \max\{i + b(S_i) | 1 \leq i \leq j\} \tag{22}$$

Equations (21) and (22) prove the lemma. □

By Lemma 4.4.3.2, there are only two possible values for the multicast time, which can be checked using Algorithm 10. Let $LB = \max\{i - 1 + b(S_i) | 1 \leq i \leq j\}$ and $UB = LB + 1$. If

$f(v_1, G, LB) = j$, then $b(v_1, U_j, G) = LB$. Otherwise, $b(v_1, U_j, G) = UB$.

Remark 4.4.3.3. *As in the case of bridge-connected graphs, it may seem that a difference of one between upper and lower bounds is not significant to study. However, when considered as a compound block of a recursive construction, based on the recurrence depth, the inaccuracy of a single time unit can add up to a major difference between broadcast/multicast times.*

4.4.4 Ring-connected graphs

Consider a graph G consisting of a cycle (ring) C with vertices $v_0, v_1, v_2, ..., v_l$ (in the cycle order), and l pairwise vertex-disjoint graphs $G_1, G_2, ..., G_l$ attached to C with the following properties:

- $v_i \in V(G_i)$, for any $1 \leq i \leq l$.
- v_0 is only adjacent to v_1 and v_l.
- For any $1 \leq i \leq l$, $1 \leq j \leq l$, $i \neq j$, an arbitrary vertex $v \in G_i$ and an arbitrary vertex $u \in G_j$, $(v, u) \notin E(G)$.

Fig. 4.6 gives an example of the above-described graph. We may denote such a graph as $G = (C, G_1, G_2, ..., G_l)$.

For a graph $G = (C, G_1, G_2, ..., G_l)$, where $C = \{v_0, v_1, v_2, ..., v_l\}$, let $S_1, S_2, ..., S_l$ be optimal broadcast schemes respecting freedom for graphs $G_1, G_2, ..., G_l$ originating from $v_1, v_2, ..., v_l$, respectively.

Note that, cycle C can be considered as two path-connected graphs: constructed by the path going from v_0, via v_1, towards v_l and the opposite direction. Hence Algorithm 10 can be applied to G if the direction of the path is specified.

Čevnik and Žerovnik (2017) considered a similar structure as a subproblem and showed upper and lower bounds on the broadcast time. Adjusting the notations, results can be summarized in the lemma below. Let A and B be defined as follows.

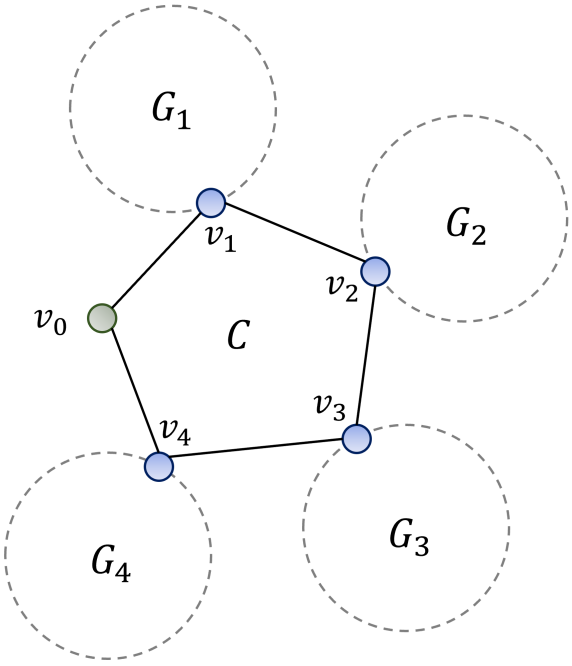


Figure 4.6: Visualization of ring-connected graphs $G = (C, G_1, G_2, G_3, G_4)$, where the cycle C is formed by v_0, v_1, v_2, v_3, v_4.

$$A = \max_{1 \leq i \leq \lceil \frac{l}{2} \rceil} \{b(S_i) + i\}$$

$$B = \max_{\frac{l}{2}+1 \leq i \leq l} \{b(S_i) + (l - i + 1)\}$$

Lemma 4.4.4.1. *The broadcast time of v_0 in G is bounded by:*

$$\begin{aligned} A \leq b(v_0, G) \leq A + 1 \qquad & A > B \\ A + 1 \leq b(v_0, G) \leq A + 2 \qquad & A = B \\ B \leq b(v_0, G) \leq B + 1 \qquad & A < B \end{aligned}$$

Again let $G = (C, G_1, G_2, ..., G_l)$ be a graph with ring-connected graphs structure, where $C = \{v_0, v_1, v_2, ..., v_l\}$, and let $S_1, S_2, ..., S_l$ be optimal broadcast schemes respecting freedom for graphs $G_1, G_2, ..., G_l$ originating from $v_1, v_2, ..., v_l$, respectively. Consider the graph PG_1, which results after copying G and removing the edge (v_0, v_l). Since the removal of (v_0, v_l) disconnects

the cycle C, PG_1 will form path-connected graphs. Similarly, the removal of (v_0, v_1) will form different path-connected graphs, namely the graph PG_2.

Assume that $b(v_0, G) = \tau$, and also assume v_0 placed the call to v_1 in the first time unit. Even though we assumed that v_1 was informed in the first time unit, the case when v_l is informed in the first time unit, instead of v_1, still needs to be considered. Let $j_1 = \text{FIXEDDEADLINEPATH}(PG_1, \tau)$, whereas $j_2 = \text{FIXEDDEADLINEPATH}(PG_2, \tau - 1)$ (Algotihm 10). Informally, j_1 is the index of the last subgraph that can be fully informed within τ time units in PG_1, whereas, j_2 is the index, in the original graph G, of the last subgraph that can be informed within $\tau - 1$ time units in PG_2. Note that procedure FIXEDDEADLINEPATH presumes that the indices of the path vertices are sorted in increasing order. Hence, the invocation of FIXEDDEADLINEPATH on PG_2 contradicts the definition of the procedure. However, a simple wrapper can easily be implemented to guarantee the desired outcome. Since an optimal broadcasting process is finished by time unit τ and v_1 is informed in the first time unit, we can claim that $j_1 \geq j_2 - 1$. If $j_1 < j_2 - 1$, then graph G_{j_2-1} would not be informed by the time unit τ.

By Lemma 4.4.4.1, in any scenario, there are two possible values for $b(v_0, G)$. We are left to check if the lower bound of the broadcast time is achievable. Moreover, we need to check the only two possible scenarios for the broadcast order of v_0: v_1 or v_l is informed in the first time unit. If the lower bound is not achievable, then we know that the broadcast time is equal to the upper bound. Formally, the procedure to calculate the broadcast time of a graph with a ring-connected graphs structure is presented in Algorithm 11.

4.4.5 Applications

Several graph families that can be represented as recursively decomposable graphs following one of the discussed compound structures were already studied in the context of broadcasting.

- *Trees* can be represented as recursively decomposable path-connected graphs. Consider any path that starts at the root (originator) and ends at any of the leaves. That path and the sub-trees attached to it form path-connected graphs. A recursive broadcasting algorithm on this representation of the tree will follow the depth-first search (DFS) order of the tree. Whereas,

Algorithm 11 Broadcast time of Ring-connected graphs

Input

1. A graph $G = (C, G_1, G_2, ..., G_l)$, where $C = \{v_0, v_1, v_2, ..., v_l\}$.

2. $S_1, S_2, ..., S_l$, where S_i is an optimal broadcast scheme respecting freedom of G_i originating from v_i.

Output Broadcast time $b(v_0, G)$.

procedure BROADCASTTIMERING(G)
  $A \leftarrow \max_{1 \leq i \leq \lceil \frac{l}{2} \rceil} \{b(S_i) + i\}$
  $B \leftarrow \max_{\frac{l}{2}+1 \leq i \leq l} \{b(S_i) + (l - i + 1)\}$
  Integer LB ▷ The lower bound on $b(v_0, G)$
  if $A > B$ **then**
    $LB = A$
  else if $B > A$ **then**
    $LB = B$
  else
    $LB = A + 1$
  end if
  $j_1 \leftarrow$ FIXEDDEADLINEPATH(PG_1, LB) (Algorithm 10) ▷ Case 1: v_1 is informed first
  $j_2 \leftarrow$ FIXEDDEADLINEPATH$(PG_2, LB - 1)$
  if $j_1 \geq j_2 - 1$ **then**
    return LB
  end if
  $j_1 \leftarrow$ FIXEDDEADLINEPATH$(PG_1, LB - 1)$ ▷ Case 2: v_l is informed first
  $j_2 \leftarrow$ FIXEDDEADLINEPATH(PG_2, LB)
  if $j_1 \geq j_2 - 1$ **then**
    return LB
  end if
  return $LB + 1$ ▷ Lower bound is not achievable
end procedure

in literature, broadcasting algorithms on trees are usually designed following the breadth-first search (BFS) order. Figure 4.7 portrays an example of this representation.

- *Unicyclic graphs* can be represented as a combination of ring-connected and path-connected graphs. The single cycle of a unicyclic graph can form the ring-connected structure, whereas, the resulting subgraphs are trees.

- *Necklace graphs* can be represented as recursively decomposable ring-connected graphs.

- *Cactus graphs* can be recursively represented as a combination of ring-connected and path-connected graphs. Indeed, a structure similar to ring-connected graphs was utilized by Čevnik

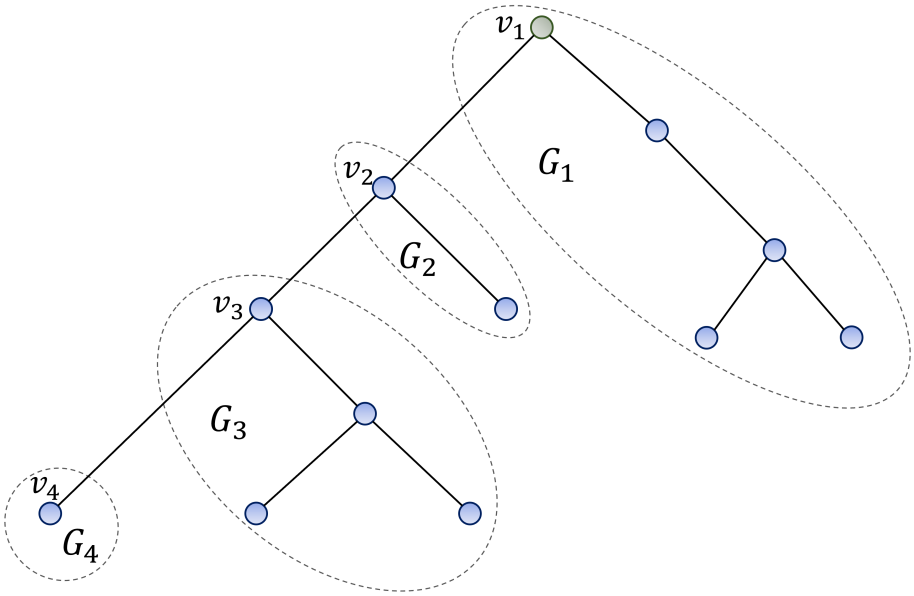


Figure 4.7: Visualization of a path-connected graphs representation of a tree, where vertices v_1, v_2, v_3, v_4 form the path and their corresponding rooted subtrees form the graphs.

and Žerovnik (2017). The authors termed it as a cycle-like component. A cycle-like component consists of a cycle and subcacti attached to all vertices of the cycle, other than the originator. They propose several algorithms to calculate a cycle-like component's full and partial broadcast times. As opposed to our proposal for ring-connected graphs, all algorithms in (Čevnik & Žerovnik, 2017) are based on the idea that attached subcacti can be replaced with a corresponding optimal broadcast scheme (trees) without any limitations. However, as per the simple example portrayed in Figure 4.8, not all optimal broadcast trees of the attached subgraphs induce an optimal broadcast scheme for the original graph. Although, in both Subfigures (a) and (b), the attached subgraph G_v is replaced with an optimal broadcast scheme, only in Subfigure (b) it results in an optimal broadcast scheme for the original graph.

This example renders Algorithms $1-5$ and Lemma 1 of (Čevnik & Žerovnik, 2017) theoretically incomplete and inapplicable to the general structure. Thus, the theoretical foundation introduced in Section 4.4 can be used to formally prove the correctness of an exact polynomial-time algorithm for k-cacti.

- Lastly, studying more compound structures adapted for broadcasting, will result in known

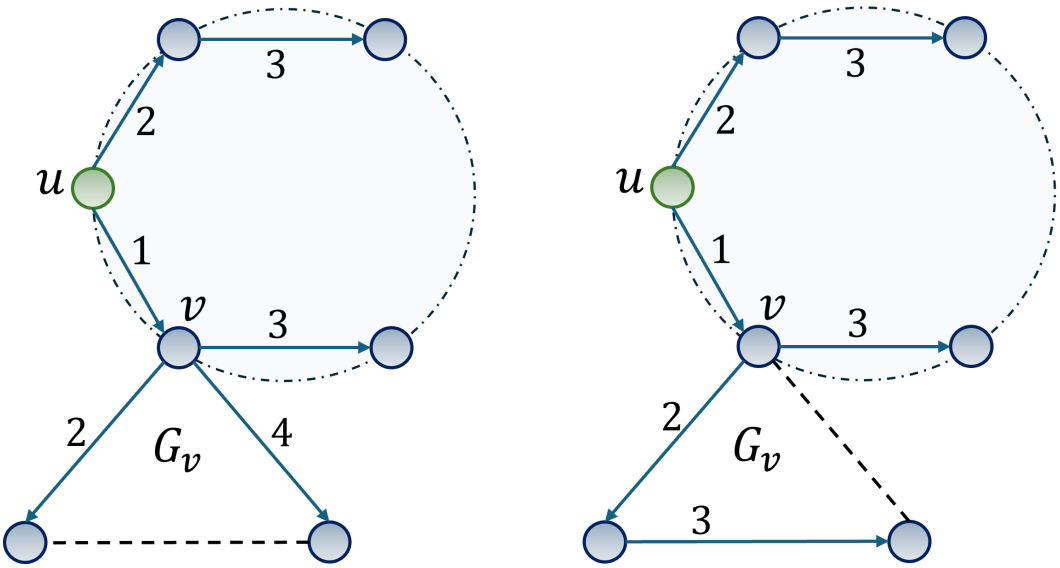


Figure 4.8: Example of a cycle and an attached graph with optimal and non-optimal broadcast schemes.

recursive representations of more graph families. For instance, chordal graphs can be represented as clique-connected graphs (clique and attached graphs). In any graph, a *vertex separator* is a set of vertices, the removal of which leaves the remaining graph disconnected. A vertex separator is minimal if it does not contain any proper subset that is a separator itself. Dirac (1961) showed that all minimal vertex separators of a chordal graph are cliques. According to Bartlett (2003), the vertex set of a chordal graph G can be partitioned into three nonempty, pairwise vertex-disjoint subsets A, B, and S, with the following properties.

- S is a clique,
- $S \cup A$ and $S \cup B$ induce chordal graphs, and
- there are no edges between A and B.

Hence, chordal graphs can, in fact, be represented as clique-connected graphs.

Chapter 5

k-Path Graphs

This chapter discusses our contributions to the broadcast time problem in k-path graphs. We present an algorithm that improves the previous best approximation ratio by a multiplicative factor of two. We also pose a new problem concerning manipulations over a sequence of numbers, which, as we believe, has not been studied previously. We demonstrate a close connection between the proposed problem and broadcasting in k-path graphs, enabling us to devise an exact polynomial-time algorithm on a subfamily of k-path graphs.

5.1 Introduction

In this section, we discuss a simple subfamily of 2-connected series-parallel graphs (Hambly & Jordan, 2004) that are usually referred to as *k-path graphs* or *melon graphs*. A k-path graph $G = (P_1, P_2, ..., P_k)$ is obtained from a pair of vertices u and v, by adding $k \geq 2$ internally vertex-disjoint paths $P_1, P_2, ..., P_k$ between u and v. Vertices u and v are called junctions of G (Figure 5.1). We assume that paths are indexed in a non-increasing order of their lengths. Formally, $l_1 \geq l_2 \geq ... \geq l_k$, where l_i is the number of vertices on path P_i (excluding u and v), for all $1 \leq i \leq k$. Note that, since we consider all input graphs to be simple, only l_k can be 0.

To have more understanding of the broadcast time problem in general graphs, it is rather important to understand the properties of graphs that make this problem difficult. One such property is the existence of intersecting cycles. Currently, k-restricted cacti and closed chains of rings are the only

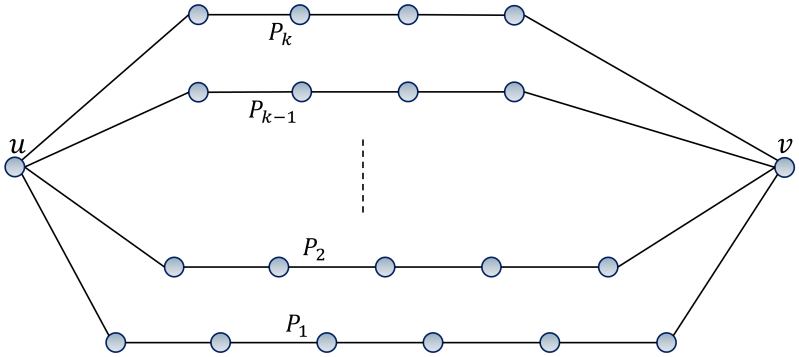


Figure 5.1: Example of a k-path graph.

graph families with intersecting cycles for which there exists an exact solution. Moreover, unlike those two families where any two cycles intersect at a single vertex, cycle intersections in k-path graphs happen at multiple vertices or edges. Hence, it is interesting to devise improved approximation algorithms or, ideally, exact algorithms for graph families such as k-path graphs, k-cycle graphs, or cactus graphs. Since k-path graphs are one of the simplest graphs containing intersecting cycles, they present an intriguing avenue for research regarding the broadcast time problem.

The broadcast time problem was researched for general series-parallel graphs as well as for subclasses of series-parallel graphs (Kortsarz & Peleg, 1995; Marathe et al., 1998). Marathe et al. (1998) prove that there is a $\mathcal{O}(\log n / \log \log n)$-approximation algorithm for the minimum broadcast time problem in graphs with bounded *treewidth*[1] (for an introduction to treewidth we refer the reader to (Bodlaender, 1998; Bodlaender & Koster, 2008)). As series-parallel graphs have a treewidth of 2, the result also applies to series-parallel graphs.

The current best results for the broadcast time problem in k-path graphs were introduced by Bhabak and Harutyunyan (2019). The authors introduce a $(4 - \epsilon)$-approximation algorithm for the broadcast time problem in general k-path graphs, for some real $\epsilon > 0$. Additionally, for some particular subclasses of k-path graphs, the authors give better approximations or optimal algorithms. The complete set of results introduced by Bhabak and Harutyunyan (2019) is presented in Table 5.1.

The main algorithm, introduced by Bhabak and Harutyunyan (2019), first counts the number of uninformed vertices on each path in each time unit. Then, based on the relation between these

[1]The treewidth of an undirected graph is an integer number which specifies, informally, how far the graph is from being a tree.

numbers decides on the order of calls placed by the junction vertices. In this thesis, we design a simple algorithm that achieves a better approximation ratio without counting the remaining lengths of the paths.

Table 5.1: Summary of known results for k-path graphs

Case	Algorithm	Result
General k-path	S_{path}	4-approximation
$l_j \geq l_{j+1} + 2$ and $k \leq l_k + 1$	S_{path}	optimal
$l_j = l_{j+1}$ and $k \leq l_k + 1$	S_{path}	optimal
$l_j = l_{j+1} + 1$ and $k \leq l_k + 1$	S_{path}	$\frac{4}{3}$-approximation
$l_j = l_{j+1} + 1$, $k \leq l_k + 1$ and u is the originator	A_{path}	$\frac{7}{6}$-approximation

5.2 Broadcasting approximation in k-path graphs

In the same paper, Bhabak and Harutyunyan (2019) also presented several lower bounds and other auxiliary results for the broadcast time problem.

Given a k-path graph G_k and junction vertex u, the authors proved the following.

Lemma 5.2.1 (Bhabak & Harutyunyan, 2019)**.** *There exists a minimum broadcast scheme from the originator u in G_k in which the shortest path P_k is informed in the first time unit.*

Similarly, given an internal vertex w, the authors proved the following.

Lemma 5.2.2 (Bhabak & Harutyunyan, 2019)**.** *There exists a minimum broadcast scheme from the originator w in G_k in which w first sends the message along a shorter path towards a junction vertex.*

5.2.1 Broadcasting from a junction vertex

Let $G = (P_1, P_2, ..., P_k)$ be a k-path graph with junction vertices u and v.

It is easy to see that in any broadcast scheme where a junction vertex is the originator, an internal (non-junction) vertex w has at most one uninformed neighbor after being informed. Thus, to describe a busy broadcast scheme, it is sufficient to specify the order of calls placed by u and v. Below is the proposed broadcasting strategy that u and v need to follow.

- As per Lemma 5.2.1, u first informs its neighbor on path P_k.
- Starting time unit 2, u informs its neighbors on longer paths first; from l_1 to l_{k-1} in the increasing order of indices.
- v receives the message via path P_k on time unit $l_k + 1$.
- Starting time unit $l_k + 2$, v informs its neighbors on shorter paths first; from l_{k-1} to l_1 in the decreasing order of indices.

Algorithm 12 formally describes the behavior of u and v in our approximation algorithm, assuming, wlog, that u is the originator. Figure 5.2 portrays the broadcast scheme introduced in Algorithm 12.

Algorithm 12 Broadcasting from a junction vertex

Input A k-path graph $G = (P_1, P_2, ..., P_k)$, junction vertices u and v, and an originator u
Output A broadcast scheme with time $b_{Alg}(u, G)$ for graph G and the originator u

1: **procedure** BROADCASTINGFROMJUNCTION
2: In the first time, unit u passes the message along the path P_k
3: **for** $2 \leq i \leq k$ **do**
4: **if** P_{i-1} contains an uninformed vertex **then**
5: u passes the message along the path P_{i-1} in time unit i
6: **end if**
7: **end for**
8: v gets the message in time unit $l_k + 1$
9: **for** $2 \leq j \leq k$ **do**
10: **if** P_{k-j+1} contains an uninformed vertex **then**
11: v passes the message along the path P_{k-j+1} in time unit $l_k + j$
12: **end if**
13: **end for**
14: **end procedure**

Complexity analysis

Note that all the basic steps of Algorithm 12 require constant time. Moreover, the order of calls placed by each vertex is predefined and does not require any computation. Meaning that the time required to inform each path can be computed in constant time. Hence, Algorithm 12 constructs a broadcast scheme in $\mathcal{O}(1)$ time and can be used to calculate the corresponding broadcast time in $\mathcal{O}(k)$ time.

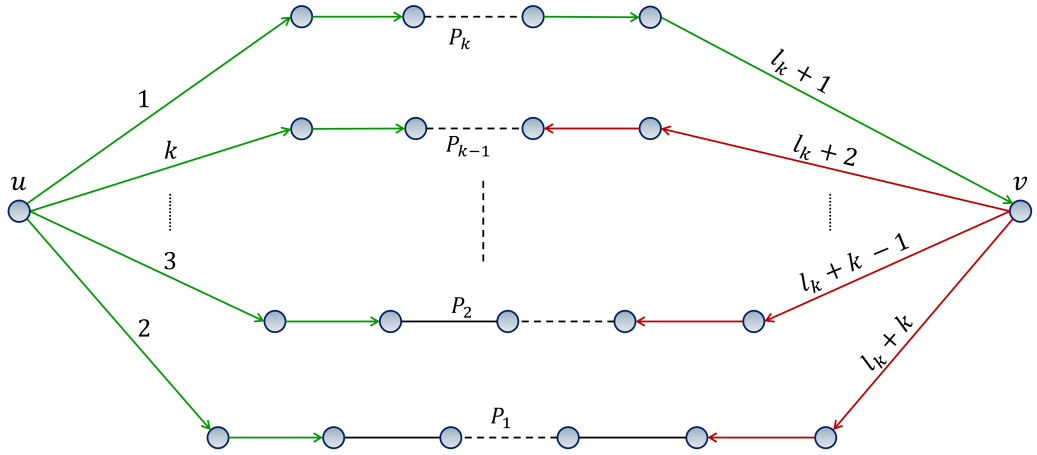


Figure 5.2: Example of the broadcast scheme described in Algorithm 12.

The approximation ratio

Theorem 5.2.1.1. *Algorithm 12 is a polynomial-time* 2*-approximation algorithm for general* k*-path graphs when the originator is a junction vertex. Moreover, when* $l_k \geq 1$*, it guarantees* $(2 - \epsilon)$*-approximation for some* $0 < \epsilon \leq 1$.

Proof. Let P_i be a path that contained an uninformed vertex before the last time unit of the broadcast scheme described in Algorithm 12. To calculate $b_{Alg}(u, G)$, we will consider three cases: vertices in P_i were informed by both u and v, P_i was informed only by u, and P_i was informed only by v.

Case 1: Only u (but not v) placed a call towards the path P_i.

According to our algorithm, u informs path P_i in time unit $i + 1$. Then, it will require $l_i - 1$ time units for the path P_i to get fully informed. Hence,

$$b_{Alg}(u, G) = l_i + i \tag{23}$$

According to Bhabak and Harutyunyan (2019), for any $1 \leq i \leq k - 1$, the following lower bound on the broadcast time holds.

$$b(u, G) \geq \left\lceil \frac{l_k + l_i + i + 1}{2} \right\rceil \geq \frac{l_k + l_i + i + 1}{2} \tag{24}$$

The approximation ratio directly follows from Equations (23) and (24).

$$\frac{b_{Alg}(u,G)}{b(u,G)} \leq \frac{l_i + i}{\frac{l_k + l_i + i + 1}{2}} < 2 \tag{25}$$

Case 2: Only v (but not u) placed a call towards the path P_i.

According to our algorithm, u informs path P_i in time unit $i+1$. Since, in this case, path P_i was informed solely by v, then we can claim that it was fully informed in time unit $i+1$ as the latest. Hence,

$$b_{Alg}(u,G) \leq i + 1 \leq k + 1 \tag{26}$$

Bhabak and Harutyunyan (2019) also proved the below lower bound.

$$b(u,G) \geq \left\lceil \frac{l_k + k + 1}{2} \right\rceil \geq \frac{l_k + k + 1}{2} \tag{27}$$

Again, the approximation ratio directly follows from Equations (26) and (27).

$$\frac{b_{Alg}(u,G)}{b(u,G)} \leq \frac{k+1}{\frac{l_k + k + 1}{2}} \leq 2 \tag{28}$$

Case 3: Both u and v placed a call towards the path P_i.

According to our algorithm, u informs path P_i in time unit $i+1$, and v informs path P_i no later than time unit $l_k + k - i + 1$. To calculate $b_{Alg}(u,G)$, we need to consider two subcases.

Subcase 3.a: $i + 1 \leq l_k + k - i + 1$.

In this case, a single vertex on path P_i is informed in each time unit between $i+1$ and $l_k + k - i + 1$. Afterward, up to 2 vertices can be informed. Hence,

$$\begin{aligned} b_{Alg}(u,G) &= l_k + k - i + \left\lceil \frac{l_i - (l_k + k - i - i)}{2} \right\rceil \\ &= \left\lceil \frac{2l_k + 2k - 2i + l_i - l_k - k + 2i}{2} \right\rceil \\ &= \left\lceil \frac{l_k + k + l_i}{2} \right\rceil \leq \frac{l_k + k + l_i + 1}{2} \end{aligned} \tag{29}$$

Subcase 3.b: $i + 1 > l_k + k - i + 1$.

Similar to the previous case,

$$\begin{aligned} b_{Alg}(u,G) &= i + \left\lceil \frac{l_i - (i - (l_k + k - i))}{2} \right\rceil \\ &= i + \left\lceil \frac{l_i - i + l_k + k - i}{2} \right\rceil \\ &= \left\lceil \frac{2i + l_i - 2i + l_k + k}{2} \right\rceil \\ &= \left\lceil \frac{l_i + l_k + k}{2} \right\rceil \leq \frac{l_k + k + l_i + 1}{2} \end{aligned} \tag{30}$$

Hence, in both subcases, we have the same upper bound on $b_{Alg}(u,G)$.

The following lower bound on the broadcast time is easy to see from Equations (24) and (27).

$$\begin{aligned} b(u,G) = \frac{b(u,G)}{2} + \frac{b(u,G)}{2} &\geq \frac{l_k + k + 1}{4} + \frac{l_k + l_i + i + 1}{4} \\ &> \frac{l_k + k + 1}{4} + \frac{l_i}{4} \\ &= \frac{l_k + k + l_i + 1}{4} \end{aligned} \tag{31}$$

From Equations (29), (30) and (31),

$$\frac{b_{Alg}(u,G)}{b(u,G)} < \frac{\frac{l_k+k+l_i+1}{2}}{\frac{l_k+k+l_i+1}{4}} = 2 \tag{32}$$

Thus, we showed that $\frac{b_{Alg}(u,G)}{b(u,G)} \leq 2$ in all possible cases. Moreover, we can see from Equations (25) and (28) that equality is only possible when $l_k = 0$.

□

5.2.2 Broadcasting from an internal vertex

Let $G = (P_1, P_2, ..., P_k)$ be a k-path graph with junction vertices u and v. Let w be a vertex on path P_m, where $1 \leq m \leq k$.

It is easy to see that in any broadcast scheme where w is the originator, any internal (non-junction) vertex other than w has at most one uninformed neighbor after being informed. Thus, in order to describe a busy broadcast scheme, it is sufficient to specify the order of calls placed by u,

v, and w.

Let d be the length of the path $\overline{wu}$, and hence, $l_m + 1 - d$ be the length of the path $\overline{wv}$. Assume, wlog, that $d \leq l_m + 1 - d$. Also, let $\tau(m) = l_m + 2 - 2d$. Note that $d \geq 1$ and $\tau(m) \geq 1$.

In the proposed algorithm, w passes the message along the shorter path towards u in the first time unit. Then, in the second time unit, w informs the vertex on path P_m towards v. Clearly, u gets informed in time unit d. Moreover, there will be $l_m + 1 - d - (d-1) = \tau(m)$ uninformed vertices on path P_m. After that, if $\tau(m) \geq l_k + 1$, then u informs its neighbor on path P_k. Otherwise, u follows the main broadcasting scheme. We let t_v denote the time unit when v is informed. The main broadcasting strategy in this case is similar to the one described in Section 5.2.1. Below is the proposed broadcasting strategy that u and v need to follow.

- As per Lemma 5.2.2, w first informs its neighbor on path $\overline{wu}$ and then its neighbor on path $\overline{wv}$.
- u receives the message in time unit d.
- In time unit $d + 1$, u passes the message along the path P_k, if $\tau(m) \geq l_k + 1$.
- After that, u informs its neighbors on longer paths first; from l_1 to l_k (or l_{k-1}) in the increasing order of indices.
- v receives the message in time unit t_v.
- Starting time unit $t_v + 1$, v informs its neighbors on shorter paths first; from l_k (or l_{k-1}) to l_1 in the decreasing order of indices.

Algorithm 13 formally describes the behavior of u, v, and w in our approximation algorithm.

Complexity analysis

Algorithm 13 has a complexity similar to the one of Algorithm 12. Hence, Algorithm 13 constructs a broadcast scheme in $\mathcal{O}(1)$ time and can be used to calculate the corresponding broadcast time in $\mathcal{O}(k)$ time.

Algorithm 13 Broadcasting from an internal vertex

Input A k-path graph $G = (P_1, P_2, ..., P_k)$, junction vertices u and v, and an originator w on path P_m

Output A broadcast scheme with time $b_{Alg}(w, G)$ for graph G and the originator w

procedure BROADCASTINGFROMINTERNAL
  In the first time unit, w passes the message along the path $\overline{wu}$
  In the second time unit, w passes the message along the path $\overline{wv}$
  if $\tau(m) \geq l_k + 1$ **then**
    u passes the message along the path P_k
    for $1 \leq i \leq k-1$ **do**
      if u has an uninformed neighbor on path P_i **then**
        u passes the message along the path P_i in time unit $d + i + 1$
      end if
    end for
    v gets the message in time unit $t_v = d + l_k + 1$
  else
    for $1 \leq i \leq k$ **do**
      if u has an uninformed neighbor on path P_i **then**
        u passes the message along the path P_i in time unit $d + i$
      end if
    end for
    v gets the message in time unit $t_v = d + \tau(m)$
  end if
  for $1 \leq j \leq k$ **do**
    if v has an uninformed neighbor on path P_j **then**
      v passes the message along the path P_j in time unit $t_v + k - j$
    end if
  end for
end procedure

The approximation ratio

Let P_i be a path that contained an uninformed vertex before the last time unit of the broadcast scheme described in Algorithm 13.

Theorem 5.2.2.1. *Algorithm 13 is a polynomial-time* 2*-approximation algorithm for general* k*-path graphs when the originator is an internal vertex and* $\tau(m) < l_k + 1$*. Moreover, when* $l_k \geq 1$*, it guarantees* $(2 - \epsilon)$*-approximation for some* $0 < \epsilon \leq 1$*.*

Proof. First, let us consider the case when $i = m$. According to our broadcast scheme, path P_m will be fully informed in time unit $1 + l_m + 1 - d - 1 = l_m + 1 - d \leq l_m$. Moreover, any optimal broadcast scheme will fully inform path P_m no earlier than $b(w, G) \geq \left\lceil \frac{l_m + l_k}{2} \right\rceil \geq \frac{l_m + l_k}{2}$. The

approximation ratio follows.

$$\frac{b_{Alg}(w,G)}{b(w,G)} \leq \frac{l_m}{\frac{l_m+l_k}{2}} = 2 \tag{33}$$

Next, we prove the theorem for any $i \neq m$, by breaking it down into three cases.

Case 1: Only u (but not v) placed a call towards the path P_i.

Path P_i requires $l_i - 1$ time units to be fully informed, after time unit $d+i$ when the first call to it happens. Hence,

$$b_{Alg}(w,G) = d + i + l_i - 1 \tag{34}$$

As per Bhabak and Harutyunyan (2019), for any $1 \leq i \leq k-1$, $i \neq m$, the following lower bound on the broadcast time holds.

$$b(w,G) \geq d + \left\lceil \frac{l_i + \tau(m) + i - 1}{2} \right\rceil \geq d + \frac{l_i + \tau(m) + i - 1}{2} \tag{35}$$

The approximation ratio directly follows from Equations (34) and (35).

$$\frac{b_{Alg}(w,G)}{b(w,G)} \leq \frac{d + l_i + i - 1}{d + \frac{l_i+\tau(m)+i-1}{2}} < 2 \tag{36}$$

□

Case 2: Only v (but not u) placed a call towards the path P_i.

According to our algorithm, u informs path P_i in time unit $d+i$. Since, in this case, path P_i was informed solely by v, then we can claim that it was fully informed in time unit $i+1$ as the latest. Hence,

$$b_{Alg}(w,G) \leq d + i \leq d + k \tag{37}$$

Bhabak and Harutyunyan (2019) also proved the following lower bound.

$$b(w,G) \geq d + \left\lceil \frac{k + \tau(m) - 1}{2} \right\rceil \geq d + \frac{k + \tau(m) - 1}{2} \tag{38}$$

Again, the approximation ratio directly follows from Equations (37) and (38).

$$\frac{b_{Alg}(w,G)}{b(w,G)} \leq \frac{d+k}{d+\frac{k+\tau(m)-1}{2}} < 2 \tag{39}$$

Case 3: Both u and v placed a call towards the path P_i.

Since $\tau(m) < l_k + 2$, then vertex v will be informed via the direct path from w, and hence, $t_v = d + \tau(m)$.

According to our algorithm, u informs path P_i in time unit $d + i$, and v informs path P_i in time unit $d + \tau(m) + k - i$.

Subcase 3.a: $d + i \leq d + \tau(m) + k - i$.

When this is the case, in every time unit between $d + i$ and $d + \tau(m) + k - i$ a single vertex on path P_i is informed. After $d + \tau(m) + k - i$ up to 2 vertices can be informed in each time unit. Hence,

$$\begin{aligned} b_{Alg}(w,G) &= d + \tau(m) + k - i - 1 + \left\lceil \frac{l_i - (d + \tau(m) + k - i - d - i)}{2} \right\rceil \\ &= \left\lceil \frac{2d + 2\tau(m) + 2k - 2i - 2 + l_i - \tau(m) - k + 2i}{2} \right\rceil \\ &= \left\lceil \frac{2d + \tau(m) + l_i + k - 2}{2} \right\rceil \leq \frac{2d + \tau(m) + l_i + k - 2}{2} \end{aligned} \tag{40}$$

Subcase 3.b: $d + i > d + \tau(m) + k - i$.

Similar to the previous case,

$$\begin{aligned} b_{Alg}(w,G) &= d + i - 1 + \left\lceil \frac{l_i - (d + i - (d + \tau(m) + k - i))}{2} \right\rceil \\ &= d + i - 1 + \left\lceil \frac{l_i - i + \tau(m) + k - i}{2} \right\rceil \\ &= \left\lceil \frac{2d + 2i - 2 + l_i - i + \tau(m) + k - i}{2} \right\rceil \\ &= \left\lceil \frac{2d + \tau(m) + l_i + k - 2}{2} \right\rceil \leq \frac{2d + \tau(m) + l_i + k - 2}{2} \end{aligned} \tag{41}$$

Hence, in both subcases, we have the same upper bound on $b_{Alg}(w,G)$.

$$b_{Alg}(w,G) \leq \frac{2d + \tau(m) + l_i + k - 2}{2} \tag{42}$$

The following lower bound on the broadcast time is easy to see from Equations (35) and (38).

$$\begin{aligned} b(w,G) &= \frac{b(w,G)}{2} + \frac{b(w,G)}{2} \geq \frac{2d + l_i + \tau(m) + i - 1}{4} + \frac{2d + k + \tau(m) - 1}{4} \\ &> \frac{2d + l_i + \tau(m) - 1}{4} + \frac{k-1}{4} = \frac{2d + l_i + k + \tau(m) - 2}{4} \end{aligned} \tag{43}$$

From Equations (42) and (43),

$$\frac{b_{Alg}(w,G)}{b(w,G)} < \frac{\frac{2d+\tau(m)+l_i+k-2}{2}}{\frac{2d+\tau(m)+l_i+k-2}{4}} = 2 \tag{44}$$

Thus, we showed that $\frac{b_{Alg}(w,G)}{b(w,G)} \leq 2$ in all possible cases. Moreover, we can see from Equation (33) that equality is only possible when $l_k = 0$.

Theorem 5.2.2.2. *Algorithm 13 is a polynomial-time 2-approximation algorithm for general k-path graphs when the originator is an internal vertex and $\tau(m) \geq l_k + 1$. Moreover, when $l_k \geq 1$, it guarantees $(2-\epsilon)$-approximation for some $0 < \epsilon \leq 1$.*

Proof. The case when $i \neq m$ is equivalent to that of Theorem 5.2.2.1

Now, we prove the theorem for any $i \neq m$, by breaking it down into three cases.

Case 1: Only u (but not v) placed a call towards the path P_i.

Path P_i requires $l_i - 1$ time units to be fully informed, after time unit $d + i + 1$ when the first call to it happens. Hence,

$$b_{Alg}(w,G) = d + i + l_i \tag{45}$$

For any $1 \leq i \leq k-1$, $i \neq m$, the following lower bound on the broadcast time holds if $\tau(m) \geq l_k + 1$ (Bhabak & Harutyunyan, 2019).

$$b(w,G) \geq d + \left\lceil \frac{l_i + l_k + i}{2} \right\rceil \geq d + \frac{l_i + l_k + i}{2} \tag{46}$$

The approximation ratio directly follows from Equations (45) and (46).

$$\frac{b_{Alg}(w,G)}{b(w,G)} \leq \frac{d+i+l_i}{d+\frac{l_i+l_k+i}{2}} < 2 \tag{47}$$

Case 2: Only v (but not u) placed a call towards the path P_i.

According to our algorithm, u informs path P_i in time unit $d+i+1$. Since, in this case, path P_i was informed solely by v, then we can claim that it was fully informed in time unit $i+1$ as the latest. Hence,

$$b_{Alg}(w,G) \leq d+i+1 \tag{48}$$

Again, the approximation ratio directly follows from Equations (48) and (47).

$$\frac{b_{Alg}(w,G)}{b(w,G)} \leq \frac{d+i+1}{d+\frac{l_i+l_k+i}{2}} < 2 \tag{49}$$

Case 3: Both u and v placed a call towards the path P_i.

Since $\tau(m) \geq l_k+2$, then vertex v will be informed via path P_k from u, and hence, $t_v = d+l_k+1$.

According to our algorithm, u informs path P_i in time unit $d+i+1$, and v informs path P_i in time unit $d+l_k+1+k-i$.

Subcase 3.a: $d+i+1 \leq d+l_k+1+k-i$.

When this is the case, in every time unit between $d+i+1$ and $d+l_k+1+k-i$ a single vertex on path P_i is informed. After $d+l_k+1+k-i$ up to 2 vertices can be informed in each time unit. Hence,

$$\begin{aligned} b_{Alg}(w,G) &= d+l_k+k-i+\left\lceil \frac{l_i-(d+l_k+1+k-i-d-i-1)}{2} \right\rceil \\ &= \left\lceil \frac{2d+2l_k+2k-2i+l_i-l_k-k+2i}{2} \right\rceil \\ &= \left\lceil \frac{2d+l_k+l_i+k}{2} \right\rceil \leq \frac{2d+l_k+l_i+k}{2} \end{aligned} \tag{50}$$

Subcase 3.b: $d+i+1 > d+l_k+1+k-i$.

Similar to the previous case,

$$\begin{aligned} b_{Alg}(w,G) &= d+i+\left\lceil \frac{l_i-(d+i+1-(d+l_k+1+k-i))}{2} \right\rceil \\ &= d+i+\left\lceil \frac{l_i-i+l_k+k-i}{2} \right\rceil \\ &= \left\lceil \frac{2d+2i+l_i-i+l_k+k-i}{2} \right\rceil \\ &= \left\lceil \frac{2d+l_k+l_i+k}{2} \right\rceil \leq \frac{2d+l_k+l_i+k}{2} \end{aligned} \tag{51}$$

Hence, in both subcases, we have the same upper bound on $b_{Alg}(w,G)$.

$$b_{Alg}(w,G) \leq \frac{2d+l_k+l_i+k}{2} \tag{52}$$

According to Bhabak and Harutyunyan (2019), the following bound holds.

$$b(w,G) \geq \left\lceil d+\frac{k+l_k}{2} \right\rceil \geq d+\frac{k+l_k}{2} \tag{53}$$

The following lower bound can be obtained from Equations (46) and (53).

$$\begin{aligned} b(w,G) &= \frac{b(w,G)}{2}+\frac{b(w,G)}{2} \geq \frac{2d+l_i+l_k+i}{4}+\frac{2d+k+l_k}{4} \\ &> \frac{2d+l_i+l_k}{4}+\frac{k}{4} = \frac{2d+l_i+l_k+k}{4} \end{aligned} \tag{54}$$

From Equations (52) and (54),

$$\frac{b_{Alg}(w,G)}{b(w,G)} < \frac{\frac{2d+l_k+l_i+k}{2}}{\frac{2d+l_i+l_k+k}{4}} = 2 \tag{55}$$

Similar to Theorem 5.2.2.1, we showed that $\frac{b_{Alg}(w,G)}{b(w,G)} \leq 2$ in all possible cases and equality is only possible when $l_k = 0$.

□

5.3 Lower bound on the broadcast time

Consider a k-path graph $G = (P_1, P_2, ..., P_k)$ with junction vertices u and v, where u be the originator. Let S be an optimal broadcast scheme such that $b(S) = b$ and u sends the message along the path P_k in the first time unit (Lemma 5.2.1). Assume T is the broadcast tree induced by S. Clearly, in T, u and v are connected by path P_k. Whereas, both subtrees rooted at u and v are formed by up to $k-1$ paths of various lengths, where each path has the vertices on a path P_i, $1 \leq i \leq k-1$, informed by u or v. We will refer to the subtrees rooted at u and v as T_u and T_v, respectively. Figure 5.3 portrays an example of such a broadcast tree.

We will first consider the case of k-path graphs where u and v are connected by an edge, i.e. $l_k = 0$.

Lemma 5.3.1. *If* $l_k = 0$, $b(u, G) \geq \left\lceil \frac{1+\sqrt{4n-7}}{2} \right\rceil$.

Proof. Clearly, both (and only) u and v will be informed after the first time unit. Assume u informs path P_i in time unit $t_i \geq 2$. By time unit b, P_i can have at most $b - t_i + 1$ informed vertices in tree T_u. Overall, paths informed by u in time units $2, 3, ..., b$ can have at most $b-1, b-2, ..., 1$ informed vertices in T_u, respectively. Thus, if we count the lengths of all paths and include u and v, the below upper bound on the number of vertices in T_u can be implied.

$$|T_u| \leq 1 + 1 + 2 + ... + (b-2) + (b-1) = 1 + \frac{b(b-1)}{2} \tag{56}$$

Analogously, the same can be claimed for v.

$$|T_v| \leq 1 + 1 + 2 + ... + (b-2) + (b-1) = 1 + \frac{b(b-1)}{2} \tag{57}$$

Since, in T, u and v are connected by an edge ($l_k = 0$), T_u and T_v contain all the vertices of T.

$$|T| = |T_u| + |T_v| = 2 + \frac{b(b-1)}{2} + \frac{b(b-1)}{2} = b^2 - b + 2 \tag{58}$$

As any broadcast tree is a spanning tree of the graph, $|T| = n$. Solving the quadratic equation

results in the below lower bound on the broadcast time.

$$\begin{aligned} b^2 - b + 2 &\geq n \\ b^2 - b + 2 - n &\geq 0 \\ b(u, G) = b &\geq \left\lceil \frac{1 + \sqrt{1 - 8 + 4n}}{2} \right\rceil = \left\lceil \frac{1 + \sqrt{4n - 7}}{2} \right\rceil \end{aligned} \tag{59}$$

□

Next, we generalize the above bound to the case when $l_k \geq 0$.

Lemma 5.3.2. *For any $l_k \geq 0$,*

$$b(u, G) \geq \left\lceil \frac{1 + l_k + \sqrt{4n - l_k^2 - 4l_k - 7}}{2} \right\rceil$$

Proof. After passing the message along path P_k in the first time unit, u follows the same restrictions as in the previous case. Thus, Equation (56) still holds. However, in this case, v will get the message in time $l_k + 1$ via path P_k and can start informing other paths starting from time unit $l_k + 2$. This implies that paths informed by v in time units $l_k + 2, l_k + 3, ..., b$ can have at most $b - l_k - 1, b - l_k - 2, ..., 1$ informed vertices in T_v, respectively. The below upper bound on the number of vertices in T_v follows.

$$\begin{aligned} |T_v| &\leq 1 + 1 + 2 + ... + (b - l_k - 2) + (b - l_k - 1) = 1 + \frac{(b - l_k)(b - l_k - 1)}{2} \\ &= \frac{b^2 - 2bl_k - b + l_k^2 + l_k + 2}{2} \end{aligned} \tag{60}$$

Unlike the previous case, here, u and v are connected by path P_k in T, which contains l_k vertices (excluding u and v). Equations 56 and 60, combined with this fact, result in the following upper bound on the order of T.

$$|T| = |T_u| + |T_v| + l_k = \frac{2b^2 - 2bl_k - 2b + l_k^2 + 3l_k + 4}{2} \tag{61}$$

Similarly, solving the below quadratic equation results in the desired lower bound.

$$\begin{aligned} \frac{2b^2 - 2bl_k - 2b + l_k^2 + 3l_k + 4}{2} &\geq n \\ 2b^2 - 2bl_k - 2b + l_k^2 + 3l_k + 4 - 2n &\geq 0 \\ b(u, G) = b &\geq \left\lceil \frac{1 + l_k + \sqrt{4n - l_k^2 - 4l_k - 7}}{2} \right\rceil \end{aligned} \tag{62}$$

□

Note that the lower bound in Lemma 5.3.2 is achieved when during the broadcasting process there are no idle vertices and all paths (other than P_k) have uninformed vertices in the last time unit.

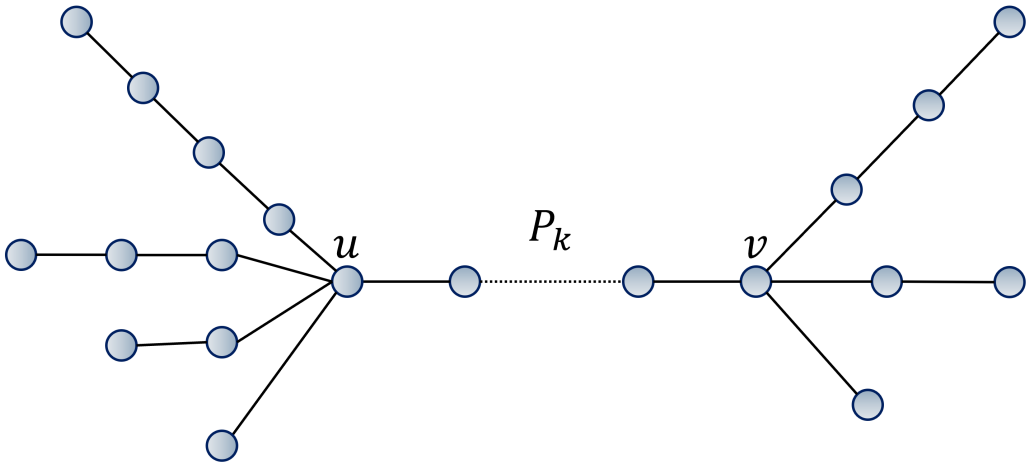


Figure 5.3: Example of a junction originator broadcast tree of a k-path graph.

5.4 3-LIST-SUB problem

For a list A, we refer to the i^{th} element of A as $A[i]$ or A_i. Unless otherwise stated, in the context of this section, we assume that indexing of lists starts from 1. Given three lists of integers A, X, and Y of the same size n, let *list subtraction* $LS(A, X, Y)$ denote a list of integers Z, such that $Z_i = A_i - X_i - Y_i$. Also, let $MLS(A, X, Y) = \max\{Z_i | 1 \leq i \leq n\}$. The goal of the 3 lists subtraction problem (3-LIST-SUB) is to find permutations X' and Y' of X and Y, that minimize $MLS(A, X', Y')$.

We formally define the general *3 list subtraction* decision problem as follows:

3 List Subtraction Problem (3-LIST-SUB):

Let A, X, and Y be lists of integers of the same size n, and k be a positive integer. Do there exist permutations X' and Y' of X and Y, respectively, that meet the following condition?

$$\max_{1 \leq i \leq n} \{A_i - X'_i - Y'_i\} \leq k$$

An example of 3-LIST-SUB instance and its solution is provided in Figure. 5.4.

	Instance						Solution				
Index	**1**	**2**	**3**	**4**	**5**		**1**	**2**	**3**	**4**	**5**
A	6	9	4	5	6		6	9	4	5	6
X	2	4	5	6	1		5	1	4	2	6
Y	1	10	3	4	1		3	10	1	4	1
Z							-2	-2	-1	-1	$\mathbf{-1}$

Figure 5.4: Example of a 3-LIST-SUB instance and its solution.

For lists A, X, and Y, we denote by $M(A, X, Y)$ the minimum value of k, for which 3-LIST-SUB can be solved on A, X, and Y. In other words, $M(A, X, Y)$ is the value of minimum MLS out of all permutations of X and Y.

It is easy to see that, regardless of the selected permutations of X and Y, the sum of the elements in the list subtraction is the same. Moreover, the theoretical minimum is achieved if all elements in the list subtraction are equal (which may not always be possible). Hence, the following lemma is trivial.

Lemma 5.4.1. *For any lists A, X, and Y of size n,*

$$M(A, X, Y) \geq \frac{\sum_{1 \leq i \leq n}(A_i - X_i - Y_i)}{n}$$

Moreover, if all the elements in the list subtraction are equal then the solution is optimal.

Particularly, we are interested in a variation of 3-LIST-SUB, where both X and Y comprise consecutive numbers, i.e. $X_i = X_{i-1} + 1$ and $Y_i = Y_{i-1} + 1$, for all $2 \leq i \leq n$.

3 Consecutive List Subtraction Problem (3-C-LIST-SUB):

Let A, X, and Y be lists of integers of the same size n, and k be a positive integer. Moreover, $X_i = X_{i-1} + 1$ and $Y_i = Y_{i-1} + 1$, for all $2 \leq i \leq n$. Do there exist permutations X' and Y' of X and Y, respectively, that meet the following condition?

$$\max_{1 \leq i \leq n} \{A_i - X'_i - Y'_i\} \leq k$$

5.4.1 Equivalency of 3-LIST-SUB instances

We will denote with $\Pi : \{1, 2, \ldots, n\} \longrightarrow \{1, 2, \ldots, n\}$, a mapping function defining the relation between the indices of a list and its permutation. We will sometimes use the notation $\Pi(X)$ to refer to the list that results after Π is applied on the indices of X. We say that a pair of permutation functions (Π_1, Π_2) is an optimal solution to a 3-LIST-SUB problem on lists A, X, and Y, if Π_1 and Π_2 applied on X and Y, respectively, result in permutation lists with minimum MLS.

Lemma 5.4.1.1. *Let A, X, Y, B, and C be lists of n integers, where X, Y, B and C consist of consecutive integers and $B_1 = C_1 = 1$. If a pair of permutation functions (Π_1, Π_2) is an optimal solution to the 3-LIST-SUB instance on A, X, and Y, then (Π_1, Π_2) is also an optimal solution to the 3-LIST-SUB instance on A, B, and C.*

Proof. Let X' and B' denote the lists that result when Π_1 is applied on X and B, respectively. More formally, $X'_j = X_i$ and $B'_j = B_i$, when $j = \Pi_1(i)$, $1 \leq i \leq n$. Similarly, let Y' and C' denote the lists that result when Π_2 is applied on Y and C, accordingly. Consider the lists $Z = LS(A, X', Y')$ and $D = LS(A, B', C')$. Since, X, Y, B, and C all comprise consecutive integers and $B_1 = C_1 = 1$, the following is easy to see for any $1 \leq i \leq n$.

- $X'_j - B'_j = X_i - B_i = (X_1 + i - 1) - i = X_1 - 1$, where $j = \Pi_1(i)$.
- $Y'_j - C'_j = Y_i - C_i = (Y_1 + i - 1) - i = Y_1 - 1$, where $j = \Pi_2(i)$.

- $D_j - Z_j = (A_j - B'_j - C'_j) - (A_j - X'_j - Y'_j) = X'_j - B'_j + Y'_j - C'_j = X_1 + Y_1 - 1$

In other words, all elements in Z and D with the same index differ by a constant. Thus, $z - d = X_1 + Y_1 - 1$, where $z = MLS(A, X', Y')$ and $d = MLS(A, B', C')$.

Assume, by contradiction, that (Π_1, Π_2) is not an optimal solution to the 3-LIST-SUB instance on A, B, and C. Let (Π'_1, Π'_2) be an optimal solution to the same instance, meaning $MLS(A, \Pi'_1(B), \Pi'_2(C)) = M(A, B, C) = d' < d$. Let $z' = MLS(A, \Pi'_1(X), \Pi'_2(Y))$ be the outcome when (Π'_1, Π'_2) is applied to the instance on A, X, and Y. Based on the above observations, $z' - d' = X_1 + Y_1 - 1$, which results in the following contradiction;

$$z' = d' + X_1 + Y_1 - 1 < d + X_1 + Y_1 - 1 < z$$

□

5.4.2 Optimal solution to restricted 3-C-LIST-SUB

Given an instance of 3-C-LIST-SUB on lists A, X, and Y of size n, where all the lists consist of consecutive integers, we are going to design optimal permutation functions. By Lemma 5.4.1.1, we can assume that $X_1 = Y_1 = 1$, since the permutation functions will also apply to other cases. We will give the optimal permutations based on the parity of n.

Odd n

Let $p = \frac{n-1}{2}$. Consider the following permutations X' and Y' of X and Y, respectively.

$$\begin{cases} X'_j = X_{2j} = 2j & \text{where } 1 \le j \le p \\ X'_{1+p+j} = X_{2j+1} = 2j + 1 & \text{where } 0 \le j \le p \end{cases} \tag{63}$$

$$\begin{cases} Y'_j = Y_{p-j+1} = p - j + 1 & \text{where } 1 \le j \le p \\ Y'_{1+p+j} = Y_{n-j} = n - j & \text{where } 0 \le j \le p \end{cases} \tag{64}$$

Theorem 5.4.2.1. *X' and Y' are optimal permutations for the* 3-C-LIST-SUB *instance on A, X, and Y of odd size n, and $M(A, X, Y) = A_1 - p - 2$.*

Proof. Following the definitions of the permutations, we are going to calculate $Z = LS(A, X', Y')$, considering each half of the list Z separately.

- For $1 \leq j \leq p$

$$Z_j = A_j - X'_j - Y'_j = (A_1 + j - 1) - (2j) - (p - j + 1) = A_1 - p - 2 \tag{65}$$

- For $0 \leq j \leq p$, $i = 1 + p + j$

$$\begin{aligned} Z_i &= A_i - X'_i - Y'_i = (A_1 + 1 + p + j - 1) - (2j + 1) - (n - j) \\ &= A_1 + p - n - 1 = A_1 + p - (2p + 1) - 1 = A_1 - p - 2 \end{aligned} \tag{66}$$

Since all elements in Z are equal, by Lemma 5.4.1, X' and Y' are optimal permutations for the 3-C-LIST-SUB instance on A, X, and Y. Moreover, $M(A, X, Y) = A_1 - p - 2$. □

Even n

Let $p = \frac{n}{2}$. Consider the following permutations X' and Y' of X and Y, respectively.

$$\begin{cases} X'_j = X_{2j} = 2j & \text{where } 1 \leq j \leq p \\ X'_{1+p+j} = X_{2j+1} = 2j + 1 & \text{where } 0 \leq j < p \end{cases} \tag{67}$$

$$\begin{cases} Y'_j = Y_{p-j+1} = p - j + 1 & \text{where } 1 \leq j \leq p \\ Y'_{1+p+j} = Y_{n-j} = n - j & \text{where } 0 \leq j < p \end{cases} \tag{68}$$

Theorem 5.4.2.2. *X' and Y' are optimal permutations for the* 3-C-LIST-SUB *instance on A, X, and Y of even size n, and $M(A, X, Y) = A_1 - p - 2$.*

Proof. Following the definitions of the permutations, we are going to calculate $Z = LS(A, X', Y')$, considering each half of the list Z separately.

- For $1 \leq j \leq p$

$$Z_j = A_j - X'_j - Y'_j = (A_1 + j - 1) - (2j) - (p - j + 1) = A_1 - p - 2 \tag{69}$$

- For $0 \leq j < p$, $i = 1 + p + j$

$$\begin{aligned} Z_i &= A_i - X'_i - Y'_i = (A_1 + 1 + p + j - 1) - (2j + 1) - (n - j) \\ &= A_1 + p - n - 1 = A_1 + p - (2p + 1) - 1 = A_1 - p - 2 \end{aligned} \tag{70}$$

Since all elements in Z are equal, by Lemma 5.4.1, X' and Y' are optimal permutations for the 3-C-LIST-SUB instance on A, X, and Y. Moreover, $M(A, X, Y) = A_1 - p - 2$. □

5.4.3 Reducing broadcast time problem on k-path graphs

Let $G = (P_1, P_2, ..., P_k)$ be a k-path graph with junction vertices u and v. Recall that we assume paths are sorted in a non-increasing order of their lengths, i.e. $l_1 \geq l_2 \geq ... \geq l_k \geq 0$.

Consider the case when u is the originator. We reduce the minimum broadcast time problem on k-path graphs to 3-C-LIST-SUB on the following lists.

(1) $A = \{l_{k-1}, l_{k-2}, \ldots, l_2, l_1\}$,

(2) $X = \{1, 2, 3, \ldots, k-1\}$,

(3) $Y = \{1, 2, 3, \ldots, k-1\}$.

Theorem 5.4.3.1. *If $b(u, G) \geq l_k + k$, the permutation functions of X and Y that optimally solve* 3-C-LIST-SUB *instance on A, X, and Y, also induce optimal broadcasting order of junctions u and v in graph G.*

Proof. By Lemma 5.2.1, u passes the message along the path P_k to v in the first time unit. Hence, v will be informed in time unit $t_v = l_k + 1$. Consider the time unit $t_v + k - 1$.

- Since v can start informing its neighbors from time unit $t_v + 1$, by time unit $t_v + k - 1$, all $k - 1$ uninformed neighbors (other than on path P_k) of v can potentially be informed. Moreover, the path that was informed on time $t_v + i$ can have $k - i$ informed vertices, for any $1 \leq i \leq k - 1$. This means v can contribute to informing $\{1, 2, \ldots, k-1\}$ vertices on some paths, depending on the order of calls placed by v.

- Similarly, u starts informing its neighbors from time unit 2 (excluding path P_k). Hence, by time unit $t_v + k - 1$, the path that was informed by u on time $1 + i$ can have $t_v + k - 1 - i$, for any $1 \leq i \leq k - 1$. Meaning u can contribute to informing $\{t_v, t_v + 1, \ldots, t_v + k - 1\}$ vertices on some paths, depending on the order of calls placed by v.

Starting from time unit $t_v + k$, neither u nor v can affect the broadcasting process. In each time unit after that, one or two vertices on each path can be informed, until the broadcasting is finished. Hence, the decisive factor of the broadcast time is the order of calls placed by u and v.

Consider the instance of 3-C-LIST-SUB on the following lists.

(1) $A = \{l_1, l_2, \ldots, l_{k-2}, l_{k-1}\}$,

(2) $B = \{1, 2, 3, \ldots, k - 1\}$,

(3) $C = \{t_v, t_v + 1, \ldots, t_v + k - 2\}$.

Let B' and C' be any permutations of B and C, respectively. Let $D = LS(A, B', C')$ and $d = MLS(A, B', C')$. Note that permutations B' and C' uniquely define the order of calls (broadcast scheme S) placed by v and u, respectively. Whereas D_i, for any $1 \leq i \leq k - 1$, corresponds to the number of uninformed vertices on path P_i in time unit $t_v + k$, if v and u followed the broadcast order defined by B' and C'. Meaning d is the highest number of uninformed vertices on any path. Assume, wlog, $d = D_j$ for some $1 \leq j \leq k - 1$. Since path P_j has the highest number of uninformed vertices d, there are two possible cases.

- If $d \geq 0$: In this case, starting from the time unit $t_v + k$, up to two vertices can be informed on path P_j. This means that the broadcasting will be over after $\lceil \frac{d}{2} \rceil$ time units, hence, the following broadcast time will be achieved.

$$b(S) = t_v + k - 1 + \left\lceil \frac{d}{2} \right\rceil = l_k + k + \left\lceil \frac{d}{2} \right\rceil \tag{71}$$

- If $d < 0$: This case implies that by time unit $t_v + k - 1$, none of the paths had any uninformed vertices, i.e. broadcast was finished and $b(S) < l_k + 1 + k - 1 = l_k + k$. Which makes this case inapplicable to this theorem. However, we can note that P_j was fully informed

- at least $\lceil \frac{d}{2} \rceil$ time units before $t_v + k - 1$, if it was informed by both u and v,
- and at most d time units before $t_v + k - 1$, if it was informed by only one of u and v.

Thus, by Equation (71), to minimize $B(S)$, d should be equal to $M(A, B, C)$.

$$b(u, G) = l_k + k + \left\lceil \frac{M(A, B, C)}{2} \right\rceil \tag{72}$$

On the other hand, by Lemma 5.4.1.1, the 3-C-LIST-SUB instance on A, B, and C has the same optimal permutations as the instance on A, X, and Y. □

The below corollary follows.

Corollary 5.4.3.2.

$$\begin{cases} b(u, G) = l_k + k + \left\lceil \frac{M(A,B,C)}{2} \right\rceil & \text{if } b(u, G) \geq l_k + k \\ l_k + k + M(A, B, C) \leq b(u, G) \leq l_k + k + \left\lceil \frac{M(A,B,C)}{2} \right\rceil & \text{if } b(u, G) < l_k + k \end{cases}$$

Another case is when a vertex w on path P_m is the originator. Let d be the length of the path $\overline{wu}$, and hence, $l_m + 1 - d$ be the length of the path $\overline{wv}$. Assume, wlog, that $d \leq l_m + 1 - d$. Also, let $\tau(m) = l_m + 2 - 2d$. Recall that when $\tau(m) < l_k + 2$, the vertex v will be informed via the direct path from w, and hence, $t_v = d + \tau(m)$. We reduce the minimum broadcast time problem on k-path graphs to 3-C-LIST-SUB on the following lists.

(1) $A = \{l_k, l_{k-1}, \ldots, l_{m+2}, l_{m+1}, l_{m-1}, l_{m-2}, \ldots, l_2, l_1\}$,

(2) $X = \{1, 2, 3, \ldots, k-1\}$,

(3) $Y = \{1, 2, 3, \ldots, k-1\}$,

(4) $B = \{1, 2, 3, \ldots, k-1\}$,

(5) $C = \{\tau(m) + 1, \tau(m) + 2, \ldots, \tau(m) + k - 1\}$.

Let $h = d + \tau(m) + k - 1$. The below claims are proved analogous to Theorem 5.4.3.1 and Corollary 5.4.3.2.

Theorem 5.4.3.3. *If $b(w, G) \geq h$, the permutation functions of X and Y that optimally solve* 3-C-LIST-SUB *instance on A, X, and Y, also induce optimal broadcasting order of junctions u and v in graph G.*

Corollary 5.4.3.4.

$$\begin{cases} b(w, G) = h + \left\lceil \frac{M(A,B,C)}{2} \right\rceil & \text{if } b(w, G) \geq h \\ h + M(A, B, C) \leq b(w, G) \leq h + \left\lceil \frac{M(A,B,C)}{2} \right\rceil & \text{if } b(w, G) < h \end{cases}$$

5.4.4 Exact broadcasting on restricted k-path graphs

Let $G = (P_1, P_2, \ldots, P_k)$ be a k-path graph, where $l_i = l_{i+1} + 1$ for any $1 \leq i \leq k - 2$. The length of path l_k is arbitrary. According to Table 5.1, the problem of finding an exact broadcasting algorithm for such k-path graphs remains open. However, combining the contributions of Sections 5.4.2 and 5.4.3 results in an exact broadcast algorithm.

By Theorem 5.4.3.1, the exact order of calls placed by u and v corresponds to the optimal permutations in the reduced 3-C-LIST-SUB instance. This means that, in the induced algorithm, both u and v should follow the reverse order of permutation lists described in Equations (63), (64), (67), and (68). An example visualizing this behavior is portrayed in Figure 5.5.

Algorithm 14 formally presents the broadcasting order.

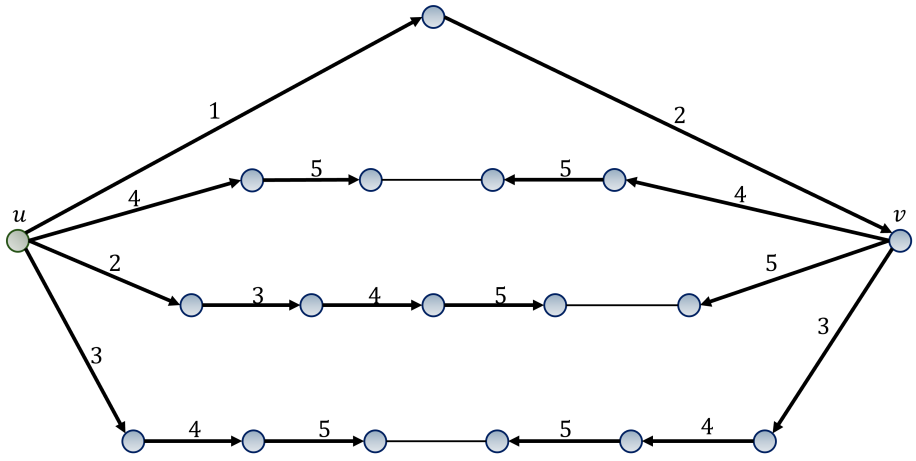


Figure 5.5: Example of the broadcast scheme in Algorithm 14.

Algorithm 14 Broadcasting in restricted k-path graphs

Input A k-path graph $G = (P_1, P_2, ..., P_k)$, junction vertices u and v, and an originator u
Output Exact broadcast scheme for graph G and the originator u

procedure EXACTBROADCASTING
    In the first time, unit u passes the message along the path P_k
    $n \leftarrow k - 1$
    if n is odd **then**
        $p \leftarrow \frac{n-1}{2}$
        for $1 \leq j \leq p$ **do**
            u informs its neighbor on path P_{k-j}, on time $n - p + j$
            v informs its neighbor on path P_{k-j}, $n - 2j + 1$ time units after being informed
        end for
        for $0 \leq j \leq p$ **do**
            u informs its neighbor on path $P_{k-j-p-1}$, on time $j + 1$
            v informs its neighbor on path $P_{k-j-p-1}$, $n - 2j$ time units after being informed
        end for
    else
        $p \leftarrow \frac{n}{2}$
        for $1 \leq j \leq p$ **do**
            u informs its neighbor on path P_{k-j}, on time $n - p + j$
            v informs its neighbor on path P_{k-j}, $n - 2j + 1$ time units after being informed
        end for
        for $0 \leq j < p$ **do**
            u informs its neighbor on path $P_{k-j-p-1}$, on time $j + 1$
            v informs its neighbor on path $P_{k-j-p-1}$, $n - 2j$ time units after being informed
        end for
    end if
end procedure

Chapter 6

Conclusion and Future Work

In an era marked by burgeoning data communication demands and the unceasing pursuit of high-performance computing, the efficiency and efficacy of information dissemination mechanisms in interconnection networks assume utmost significance. As a foundational primitive for information dissemination, the study and refinement of broadcasting mechanisms emerge as an essential research direction. Broadcasting is the problem of distributing a message from a single source throughout the network to inform all members as quickly as possible. The goal of this research is to contribute theoretical frameworks, algorithmic intricacies, and empirical investigations in the field of broadcasting, with a particular focus on its applications within interconnection network topologies comprising known structural decompositions.

Broadcasting is formally modeled using a graph $G(V, E)$, where the originator $v \in V$ has the message at the beginning and tries to pass it to all members of V. Being NP-complete in arbitrary graphs, this problem has remained interesting yet challenging for researchers in recent years. The problem remains NP-hard even for some restricted graph families, such as bounded degree graphs (Dinneen, 1994) and 3-regular planar graphs (Middendorf, 1993). Moreover, Elkin and Kortsarz (2005a) showed that it is NP-hard to design a broadcasting algorithm that guarantees $3 - \epsilon$ approximation ratio for any $\epsilon > 0$. The current best approximation ratio of $\mathcal{O}(\frac{\log n}{\log \log n})$ for broadcasting was introduced by Elkin and Kortsarz (2006). To date, exact polynomial-time solutions are known only for a limited number of graph families, such as trees, unicyclic graphs, trees of cycles, hypercubes, grids, etc. Most of such families either have simple structures with no complicated cycle

intersections or exhibit properties, such as symmetry, regularity, and high connectivity, that make the broadcasting problem polynomial-time solvable. One of the established and essential directions concerning broadcasting research is to identify structural properties of graphs that make the problem NP-hard to solve or approximate within a constant ratio.

Chapter 3 addresses broadcasting on graphs where the vertex set can be partitioned into vertex-disjoint cliques and independent sets. Clearly, any arbitrary graph can be represented as a union of several cliques and independent sets. In the worst-case scenario, any connected graph can be divided into $|V| - 1$ cliques and no independent sets. Hence, it is interesting to study this property of graphs. We start with one of the simplest cases of this partitioning, split graphs. The vertex set of a split graph can be partitioned into a clique and an independent set. Jansen and Müller (1995) proved that deciding whether the broadcast time from multiple originators in split graphs is equal to 2 is NP-complete, making split graphs an intriguing subject for broadcasting research. We proposed a polynomial-time 2-approximation algorithm for broadcasting on an arbitrary split graph. Additionally, we analyzed several key properties of an optimal broadcast scheme on split graphs, assisting in designing a strategy for generating optimal or near-optimal broadcast schemes. The optimality of the generated algorithm depends on the solution of the newly proposed minimum cost proper star-matching problem. Using this strategy, we further devised a heuristic for broadcasting on split graphs, which eventually resulted in a successful empirical outcome. Since we were able to broadcast on split graphs within an approximation ratio of 2, it was natural to consider graphs with more complex alternatives of the same partitioning. As a natural generalization of split graphs, (k, l)-graphs were the next topology to study. The vertex set of an arbitrary (k, l)-graph can be partitioned into k independent sets and l cliques. All partitions are pairwise vertex-disjoint. Recall that any connected graph can be represented as $(0, |V| - 1)$-graph. Utilizing the strategies that we developed for split graphs, a broadcasting algorithm was devised for (k, l)-graphs. In comparison with the heuristic on split graphs and lower bounds of broadcast time, the algorithm was empirically shown to achieve good results. One of the main problems that remains open is to theoretically prove an approximation ratio for the (k, l)-graph broadcasting algorithm, or to utilize known strategies applicable to split graphs to devise a new approximation algorithm. Another open problem is to solve or prove the NP-hardness of the proposed minimum cost star-matching problem.

Next, Chapter 4 focuses on a property shared among various graph families. Our goal was to provide insight into the broadcast time problem on topologies that display recursive structures. First, we proposed an exact polynomial-time broadcasting algorithm for closed chains of rings. These graphs are an extension of necklace graphs, where the first and the last cycles are also connected at a vertex. Later in the chapter, we began designing a theoretical framework to aid any future research concerning broadcasting on recursively decomposable graphs. We initiate the study of a lazy approach to broadcasting. The vast majority of research work in the area of broadcasting only examines busy broadcast schemes, which are the schemes that avoid vertex idling. On the contrary, we propose a strategy to consider broadcast schemes that are as lazy as possible, measured by the degree of freedom of the originator. The degree of freedom of a vertex in a broadcast scheme is the first time unit when the vertex is idle after receiving the message. Our aim was to analyze several compounding structures, that can be employed to represent various recursive structures, using optimal broadcast schemes with maximum laziness. For some recursive structures, we established a theoretical foundation and devised several algorithms that can be used to find the broadcast time, the multicast time, or the largest subgraph that can be informed within a fixed deadline. We studied compounding strategies where arbitrary subgraphs are connected via a topology of an edge, a path, or a ring. Toward the end of the chapter, we presented several graph families that embody the mentioned compounding structures, such as trees, unicyclic graphs, necklace graphs, and cacti. Moreover, we showed that the ring-connected graphs structure can be used to improve and formally prove some of the subprocedures used for the known exact algorithm on k-cacti. Two of the most important directions to continue studying this approach involve extending the proposed theoretical framework to other compounding methods and considering more general variations of lazy broadcasting. As per the argument that we briefly brought up in Section 4.4.5, chordal graphs can be represented as clique-connected graphs. Since classical broadcasting on chordal graphs is NP-complete (Jansen & Müller, 1995), the family of chordal graphs is a perfect candidate to investigate. Additionally, we believe that it may be possible, yet complicated, to explore considerations beyond just the first time unit when the originator was idle.

Lastly, Chapter 5 concentrates on one of the properties of graphs that evidently makes the broadcast time problem difficult: the existence of intersecting cycles. Almost all graph families for which

an exact polynomial-time broadcasting algorithm is known either do not contain intersecting cycles or any two cycles intersect at a single vertex. Increasing the complexity of cycle intersections is one of the ways to delineate the boundary between graph families where the broadcast time problem is solvable or not. The subject of interest in this chapter is the family of k-path graphs. A k-path graph consists of k internally vertex-disjoint paths of arbitrary lengths, such that all paths meet at the endpoints (called junctions). First, we devised a polynomial-time 2-approximation algorithm for broadcasting on k-path graphs that improves the previous results of Bhabak and Harutyunyan (2019) by a multiplicative factor of two. Further, we introduced a new problem that we refer to as the Three List Subtraction Problem (3-LIST-SUB). The goal of the problem is to find permutations of two of the three given lists, such that element-wise subtraction of these two lists from the first list results in a list with the lowest maximum element value. We introduced an optimal solution to a restricted subproblem to 3-LIST-SUB, helping us to devise an exact broadcasting algorithm on restricted k-path graphs with consecutive path lengths. Additionally, for k-path graphs with significantly larger broadcast times of the junctions, we showed that the length of the shortest path does not affect the broadcast order of the junctions. Looking ahead, the problem of designing a polynomial-time exact broadcasting algorithm for arbitrary k-path graphs remains open. Given that the introduced 2-approximation ratio for k-path graphs is less than the known inapproximability lower bound of $3 - \epsilon$, it is important to consider other, potentially more complicated graph families that comprise intersecting cycles.

Some of the results of this thesis are presented in (Harutyunyan & Hovhannisyan, 2023a, 2023b, 2023c, 2023d; Harutyunyan, Hovhannisyan, & Maraachlian, 2023). Results of collaboration with other researchers are presented in (Harutyunyan et al., 2022).

References

Ahlswede, R., Gargano, L., Haroutunian, H., & Khachatrian, L. H. (1996). Fault-tolerant minimum broadcast networks. *Networks*, *27*(4), 293–307.

Ahlswede, R., Haroutunian, H., & Khachatrian, L. H. (1994). Messy broadcasting in networks. In *Communications and Cryptography* (pp. 13–24). Springer.

Albin, N., Kottegoda, K., & Poggi-Corradini, P. (2020). Spanning tree modulus for secure broadcast games. *Networks*, *76*(3), 350–365. doi: 10.1002/net.21945

Al Faisal, F., Rahman, M. H., & Inoguchi, Y. (2017). A new power efficient high performance interconnection network for many-core processors. *Journal of Parallel and Distributed Computing*, *101*, 92–102.

Alon, N., Bar-Noy, A., Linial, N., & Peleg, D. (1991). A lower bound for radio broadcast. *Journal of Computer and System Sciences*, *43*(2), 290–298.

Averbuch, A., Peeri, I., & Roditty, Y. (2017). New upper bound on m-time-relaxed k-broadcast graphs. *Networks*, *70*(2), 72–78.

Averbuch, A., Shabtai, R. H., & Roditty, Y. (2014). Efficient construction of broadcast graphs. *Discrete Applied Mathematics*, *171*, 9–14.

Bakhtar, S., Gholami, S., & Harutyunyan, H. A. (2020). A new metric to evaluate communities in social networks using geodesic distance. In *International Conference on Computational Data and Social Networks (CSoNet)* (pp. 202–216).

Bar-Noy, A., Bruck, J., Ho, C.-T., Kipnis, S., & Schieber, B. (1995). Computing global combine operations in the multiport postal model. *IEEE Transactions on Parallel and Distributed Systems*, *6*(8), 896–900.

Bar-Noy, A., Guha, S., Naor, J., & Schieber, B. (1998). Multicasting in heterogeneous networks. In *Proceedings of the thirtieth annual ACM Symposium on Theory of computing* (pp. 448–453).

Bar-Noy, A., & Kipnis, S. (1994). Broadcasting multiple messages in simultaneous send/receive systems. *Discrete Applied Mathematics*, *55*(2), 95–105.

Bar-Noy, A., Kipnis, S., & Schieber, B. (2000). Optimal multiple message broadcasting in telephone-like communication systems. *Discrete Applied Mathematics*, *100*(1-2), 1–15.

Barsky, G., Grigoryan, H., & Harutyunyan, H. A. (2014). Tight lower bounds on broadcast function for n= 24 and 25. *Discrete Applied Mathematics*, *175*, 109–114.

Bartlett, P. (2003). Undirected graphical models: Chordal graphs, decomposable graphs, junction trees, and factorizations. *Lecture Notes available online at the address http://www. stat. berkeley. edu/bartlett/courses/241A-spring2007/graphnotes. pdf*.

Bar-Yehuda, R., Goldreich, O., & Itai, A. (1991). Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection. *Distributed Computing*, *5*(2), 67–71.

Beauquier, B., Perennes, S., & Delmas, O. (2001). Tight bounds for broadcasting in the linear cost model. *Journal of Interconnection Networks*, *2*(02), 175–188.

Beier, R., & Sibeyn, J. F. (2000). A powerful heuristic for telephone gossiping. In *Proceedings of the 7th International Colloquium on Structural Information and Communication Complexity (SIROCCO)* (pp. 17–35).

Belik, I. (2016). The analysis of split graphs in social networks based on the k-cardinality assignment problem. *International Journal of Network Science*, *1*(1), 53–62.

Belmonte, R., Kim, E. J., Lampis, M., Mitsou, V., Otachi, Y., & Sikora, F. (2021). Token sliding on split graphs. *Theory of Computing Systems*, *65*, 662–686.

Bender, E. A., Richmond, L. B., & Wormald, N. C. (1985). Almost all chordal graphs split. *Journal of the Australian Mathematical Society*, *38*(2), 214–221.

Ben-Moshe, B., Dvir, A., Segal, M., & Tamir, A. (2012). Centdian computation in cactus graphs. *J. Graph Algorithms Appl.*, *16*(2), 199–224.

Bermond, J.-C., Fraigniaud, P., & Peters, J. G. (1995). Antepenultimate broadcasting. *Networks*, *26*(3), 125–137.

Bermond, J.-C., Harutyunyan, H. A., Liestman, A. L., & Pérennes, S. (1997). A note on the dimensionality of modified knödel graphs. *Int. J. Found. Comput. Sci.*, *8*, 109-116.

Bermond, J.-C., Hell, P., Liestman, A. L., & Peters, J. G. (1992a, 02). Broadcasting in bounded degree graphs. *SIAM Journal on Discrete Mathematics*, *5*, 10-24.

Bermond, J.-C., Hell, P., Liestman, A. L., & Peters, J. G. (1992b). Sparse broadcast graphs. *Discrete Applied Mathematics*, *36*(2), 97–130.

Bermond, J.-C., & Peyrat, C. (1988). Broascasting in de Bruijn networks. *Congressus Numerantium*, *66*, 283–292.

Berthome, P., Ferreira, A., & Perennes, S. (1996). Optimal information dissemination in star and pancake networks. *IEEE Transactions on Parallel and Distributed Systems*, *7*(12), 1292–1300.

Bertossi, A. A. (1984). Dominating sets for split and bipartite graphs. *Information Processing Letters*, *19*(1), 37–40.

Bertsekas, D. P., Özveren, C., Stamoulis, G. D., Tseng, P., & Tsitsiklis, J. N. (1991). Optimal communication algorithms for hypercubes. *Journal of Parallel and Distributed computing*, *11*(4), 263–275.

Bhabak, P. (2014). *Approximation algorithms for broadcasting in simple graphs with intersecting cycles* (Unpublished doctoral dissertation). Concordia University.

Bhabak, P., & Harutyunyan, H. A. (2014). Broadcast problem in hypercube of trees. In *International Workshop on Frontiers in Algorithmics (FAW)* (pp. 1–12).

Bhabak, P., & Harutyunyan, H. A. (2015). Constant Approximation for Broadcasting in k-cycle Graph. In *Conference on Algorithms and Discrete Applied Mathematics (CALDAM)* (pp. 21–32).

Bhabak, P., & Harutyunyan, H. A. (2019). Approximation Algorithm for the Broadcast Time in k-Path Graph. *Journal of Interconnection Networks*, *19*(04), 1950006.

Bhabak, P., & Harutyunyan, H. A. (2022). Approximation Algorithms in Graphs with Known Broadcast Time of the Base Graph. In *Conference on Algorithms and Discrete Applied Mathematics* (pp. 305–316).

Bhabak, P., Harutyunyan, H. A., & Kropf, P. (2017). Efficient Broadcasting Algorithm in Harary-like Networks. In *2017 46th International Conference on Parallel Processing Workshops (ICPPW)* (p. 162-170).

Bhabak, P., Harutyunyan, H. A., & Tanna, S. (2014). Broadcasting in Harary-Like Graphs. In *2014 IEEE 17th International Conference on Computational Science and Engineering (CSE)* (p. 1269-1276).

Bienstock, D. (1988). Broadcasting with random faults. *Discrete Applied Mathematics*, *20*(1), 1–7.

Biggs, N. L., Damerell, R. M., & Sands, D. A. (1972). Recursive families of graphs. *Journal of Combinatorial Theory, Series B*, *12*(2), 123–131.

Birchler, B. D., Esfahanian, A.-H., & Torng, E. K. (1996). Information dissemination in restricted routing networks. In *Proceedings of the international symposium on combinatorics and applications* (pp. 33–44).

Bodlaender, H. L. (1998). A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, *209*(1-2), 1–45.

Bodlaender, H. L., Kloks, T., Tan, R. B., & van Leeuwen, J. (2000). λ-coloring of graphs. In *17th Annual Symposium on Theoretical Aspects of Computer Science (STACS)* (pp. 395–406).

Bodlaender, H. L., & Koster, A. M. (2008). Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, *51*(3), 255–269.

Brandstädt, A. (1996). Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics*, *152*(1-3), 47–54.

Brandstädt, A. (1998). Corrigendum: Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics*, *186*, 295–295.

Brandstädt, A., & Szymczak, T. (1998). The complexity of some problems related to graph 3-colorability. *Discrete Applied Mathematics*, *89*(1-3), 59–73.

Bruck, J., Cypher, R., & Ho, C.-t. (1992, 12). Multiple message broadcasting with generalized fibonacci trees. In *Proceedings of the Fourth IEEE Symposium on Parallel and Distributed Processing (IPDPS).*

Bruck, J., & Ho, C.-T. (1993). Efficient global combine operations in multi-port message-passing systems. *Parallel Processing Letters*, *3*(04), 335–346.

Cane, V. R. (1966). A note on the size of epidemics and the number of people hearing a rumour. *Journal of the Royal Statistical Society: Series B (Methodological)*, *28*(3), 487–490.

Chau, S.-C., & Liestman, A. L. (1985). Constructing minimal broadcast networks. *J. Combin. Inform. Syst. Sci*, *10*, 110–122.

Chen, M.-S., Shin, K. G., & Kandlur, D. D. (1990). Addressing, routing, and broadcasting in hexagonal mesh multiprocessors. *IEEE Transactions on Computers*, *39*(1), 10–18.

Chinn, P., Hedetniemi, S., & Mitchell, S. (1979). Multiple-message broadcasting in complete graphs. In *Proceedings of the 10th SE Conference on Combinatorics, Graph Theory and Computing (SEICCGTC). Utilitas Math. Winnipeg* (pp. 251–260).

Chlamtac, I., & Kutten, S. (1985). On broadcasting in radio networks-problem analysis and protocol design. *IEEE Transactions on Communications*, *33*(12), 1240–1246.

Chlamtac, I., & Kutten, S. (1987). Tree-based broadcasting in multihop radio networks. *IEEE Transactions on Computers*, *100*(10), 1209–1223.

Collins, K. L., & Trenk, A. N. (2017a). Finding balance: split graphs and related classes. *arXiv preprint arXiv:1706.03092*.

Collins, K. L., & Trenk, A. N. (2017b). Finding balance: split graphs and related classes. *arXiv preprint arXiv:1706.03092*.

Comellas, F., Harutyunyan, H. A., & Liestman, A. L. (2003). Messy broadcasting in multidimensional directed tori. *Journal of Interconnection Networks*, *4*(01), 37–51.

Comellas, F., & Hell, P. (2003). Broadcasting in generalized chordal rings. *Networks*, *42*(3), 123–134.

Comellas, F., & Mitjana, M. (1998). Broadcasting in cycle prefix digraphs. *Discrete applied mathematics*, *83*(1-3), 31–39.

Crescenzi, P., Fraigniaud, P., Halldorsson, M., Harutyunyan, H. A., Pierucci, C., Pietracaprina, A., & Pucci, G. (2016). On the complexity of the shortest-path broadcast problem. *Discrete Applied Mathematics*, *199*, 101–109.

Daley, D. J., & Kendall, D. G. (1964). Epidemics and rumours. *Nature*, *204*(4963), 1118–1118.

Daley, D. J., & Kendall, D. G. (1965). Stochastic rumours. *IMA Journal of Applied Mathematics*, *1*(1), 42–55.

de Sousa, A., Gallo, G., Gutierrez, S., Robledo, F., Rodríguez-Bocca, P., & Romero, P. (2018). Heuristics for the minimum broadcast time. *Electronic Notes in Discrete Mathematics*, *69*, 165–172.

de Sousa, A., Gallo, G., Gutiérrez, S., Robledo, F., Rodríguez-Bocca, P., & Romero, P. (2020). A tree-block decomposition-based heuristic for the minimum broadcast time. *International Journal of Metaheuristics*, *7*(4), 379–401.

Dessmark, A., Lingas, A., Olsson, H., & Yamamoto, H. (1998). Optimal Broadcasting in Almost Trees and Partial k-trees. In *15th Annual Symposium on Theoretical Aspects of Computer Science (STACS)* (pp. 432–443). Springer.

Dietzfelbinger, M. (2004). Gossiping and broadcasting versus computing functions in networks. *Discrete Applied Mathematics*, *137*(2), 127–153.

Diks, K., & Pelc, A. (1996). Broadcasting with universal lists. *Networks*, *27*(3), 183–196.

Dinneen, M. J. (1994). The complexity of broadcasting in bounded-degree networks. *arXiv preprint math/9411222*.

Dinneen, M. J., Fellows, M. R., & Faber, V. (1991). Algebraic constructions of efficient broadcast networks. In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes (AAECC)* (pp. 152–158).

Dirac, G. A. (1961). On rigid circuit graphs. In *Abhandlungen aus dem mathematischen seminar der universität hamburg* (Vol. 25, pp. 71–76).

Elenbogen, B., & Fink, J. F. (2007). Distance distributions for graphs modeling computer networks. *Discrete applied mathematics*, *155*(18), 2612–2624.

Elkin, M., & Kortsarz, G. (2002). Combinatorial logarithmic approximation algorithm for directed telephone broadcast problem. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing* (pp. 438–447).

Elkin, M., & Kortsarz, G. (2004). Polylogarithmic inapproximability of the radio broadcast problem. In *International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)* (pp. 105–116).

Elkin, M., & Kortsarz, G. (2005a). A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem. *SIAM Journal on Computing*, *35*(3), 672–689.

Elkin, M., & Kortsarz, G. (2005b). Polylogarithmic additive inapproximability of the radio broadcast problem. *SIAM Journal on Discrete Mathematics*, *19*(4), 881–899.

Elkin, M., & Kortsarz, G. (2006). Sublogarithmic approximation for telephone multicast. *Journal of Computer and System Sciences*, *72*(4), 648–659.

Eschen, E. M., & Wang, X. (2014). Algorithms for unipolar and generalized split graphs. *Discrete Applied Mathematics*, *162*, 195–201.

Even, S., & Monien, B. (1989). On the number of rounds necessary to disseminate information. In *Proceedings of the first annual ACM symposium on Parallel algorithms and architectures* (pp. 318–327).

Farley, A. M. (1979). Minimal broadcast networks. *Networks*, *9*(4), 313–332.

Farley, A. M. (1980a). Broadcast time in communication networks. *SIAM Journal on Applied Mathematics*, *39*(2), 385–390.

Farley, A. M. (1980b). Minimum-time line broadcast networks. *Networks*, *10*(1), 59–70.

Farley, A. M. (2004). Minimal path broadcast networks. *Networks*, *43*(2), 61–70.

Farley, A. M., Fragopoulou, P., Krumme, D., Proskurowski, A., & Richards, D. (2000). Multi-source spanning tree problems. *Journal of Interconnection Networks*, *01*(01), 61-71.

Farley, A. M., & Hedetniemi, S. T. (1978). Broadcasting in grid graphs. In *Proceedings 9th S.E. Conference on Combinatorics, Graph Theory, and Computing, Utilitas Mathematica* (pp. 275–288).

Farley, A. M., Hedetniemi, S. T., Mitchell, S., & Proskurowski, A. (1979). Minimum broadcast graphs. *Discrete Mathematics*, *25*(2), 189–193.

Farley, A. M., & Proskurowski, A. (1981a). Broadcasting in trees with multiple originators. *SIAM Journal on Algebraic Discrete Methods*, *2*(4), 381–386.

Farley, A. M., & Proskurowski, A. (1981b). Broadcasting in trees with multiple originators. *SIAM Journal on Algebraic Discrete Methods*, *2*(4), 381-386.

Farley, A. M., & Proskurowski, A. (1994). Bounded-call broadcasting. *Discrete Applied Mathematics*, *53*(1-3), 37–53.

Feldmann, R., Hromkovič, J., Madhavapeddy, S., Monien, B., & Mysliwietz, P. (1994). Optimal algorithms for dissemination of information in generalized communication modes. *Discrete*

Applied Mathematics, *53*(1-3), 55–78.

Foldes, S., & Hammer, P. L. (1977). Split graphs. In *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing (SECCGTC)* (Vol. XIX, p. 311-–315).

Fomin, F. V., Fraigniaud, P., & Golovach, P. A. (2023). Parameterized complexity of broadcasting in graphs. *arXiv preprint arXiv:2306.01536*.

Fraigniaud, P. (2001a). Approximation algorithms for minimum-time broadcast under the vertex-disjoint paths mode. In *European Symposium on Algorithms (ESA)* (pp. 440–451).

Fraigniaud, P. (2001b). Minimum-time broadcast under edge-disjoint paths modes. In *International Conference on Fun with Algorithms (FUN).*

Fraigniaud, P., & Lazard, E. (1994). Methods and problems of communication in usual networks. *Discrete Applied Mathematics*, *53*(1), 79-133.

Fraigniaud, P., & Peters, J. G. (2001). Minimum linear gossip graphs and maximal linear (δ, k)-gossip graphs. *Networks*, *38*(3), 150-162.

Fraigniaud, P., & Vial, S. (1996). Approximation algorithms for information dissemination problems. In *Proceedings of 1996 IEEE Second International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP)* (pp. 155–162).

Fraigniaud, P., & Vial, S. (1997). Approximation algorithms for broadcasting and gossiping. *Journal of Parallel and Distributed Computing*, *43*(1), 47–55.

Fraigniaud, P., & Vial, S. (1999). Comparison of heuristics for one-to-all and all-to-all communications in partial meshes. *Parallel Processing Letters*, *9*(01), 9–20.

Fulkerson, D. R., & Ford, L. R. (1962). *Flows in networks*. Princeton University Press Princeton.

Garey, M. R., & Johnson, D. S. (1983). *Computers and intractability. A guide to the theory of NP-Completeness.*

Gargano, L., & Vaccaro, U. (1992). Minimum time broadcast networks tolerating a logarithmic number of faults. *SIAM Journal on Discrete Mathematics*, *5*(2), 178–198.

Gholami, S., & Harutyunyan, H. A. (2022a). Broadcast Graphs with Nodes of Limited Memory. In *13th International Conference on Complex Networks (CompleNet).*

Gholami, S., & Harutyunyan, H. A. (2022b). Fully-adaptive model for broadcasting with universal

lists. In *24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* (p. 92-99).

Gholami, S., & Harutyunyan, H. A. (2022c). HUB-GA: A Heuristic for Universal lists Broadcasting using Genetic Algorithm. *Journal of Communications and Networks*.

Gholami, S., Harutyunyan, H. A., & Maraachlian, E. (2023). Optimal broadcasting in fully connected trees. *Journal of Interconnection Networks*, *23*(01), 2150037.

Gitman, I., Van Slyke, R., & Frank, H. (1976). Routing in packet-switching broadcast radio networks. *IEEE Transactions on Communications*, *24*(8), 926–930.

Golumbic, M. C. (2004). *Algorithmic graph theory and perfect graphs*. Elsevier.

Grigni, M., & Peleg, D. (1991). Tight bounds on mimimum broadcast networks. *SIAM Journal on Discrete Mathematics*, *4*(2), 207–222.

Grigoryan, H., & Harutyunyan, H. A. (2014). Diametral broadcast graphs. *Discrete Applied Mathematics*, *171*, 53–59.

Hambly, B. M., & Jordan, J. (2004). A random hierarchical lattice: the series-parallel graph and its properties. *Advances in Applied Probability*, *36*(3), 824–838.

Hammer, P. (1977). Aggregation of inequalities in integer programming. *Ann. Discrete Math*, *1*, 145–162.

Harutyunyan, H. A. (2000). Multiple broadcasting in modified Knödel graphs. In *7th International Colloquium on Structural Information and Communication Complexity (SIROCCO)* (pp. 157–166).

Harutyunyan, H. A. (2006a). Minimum multiple message broadcast graphs. *Networks*, *47*(4), 218-224.

Harutyunyan, H. A. (2006b). Minimum multiple message broadcast graphs. *Networks*, *47*(4), 218–224.

Harutyunyan, H. A. (2008). An efficient vertex addition method for broadcast networks. *Internet Mathematics*, *5*(3), 211–225.

Harutyunyan, H. A., & Hovhannisyan, N. (2023a). Broadcasting in interconnection networks based on node partitioning. In *2023 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)* (pp. 234–239).

Harutyunyan, H. A., & Hovhannisyan, N. (2023b). Broadcasting in split graphs. In *International Conference on Algorithms and Complexity (CIAC)* (pp. 278–292).

Harutyunyan, H. A., & Hovhannisyan, N. (2023c). Efficient heuristic for broadcasting in chordal networks. In *International Conference on Advanced Information Networking and Applications (AINA)* (pp. 332–345).

Harutyunyan, H. A., & Hovhannisyan, N. (2023d). Improved approximation for broadcasting in k-path graphs. In *International Conference on Combinatorial Optimization and Applications (COCOA)* (pp. 111–122).

Harutyunyan, H. A., Hovhannisyan, N., & Magithiya, R. (2022). Deep heuristic for broadcasting in arbitrary networks. In *21st International Symposium on Parallel and Distributed Computing (ISPDC)* (p. 1-8). Los Alamitos, CA, USA: IEEE Computer Society. (Best Paper Award)

Harutyunyan, H. A., Hovhannisyan, N., & Maraachlian, E. (2023). Broadcasting in chains of rings. In *2023 Fourteenth International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 506–511).

Harutyunyan, H. A., & Jimborean, C. (2014). New heuristic for message broadcasting in networks. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications (AINA)* (pp. 517–524).

Harutyunyan, H. A., & Kamali, S. (2008a). Broadcasting in weighted-vertex graphs. In *IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)* (pp. 301–307).

Harutyunyan, H. A., & Kamali, S. (2008b). Efficient broadcasting in networks with weighted nodes. In *14th IEEE International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 879–884).

Harutyunyan, H. A., & Kamali, S. (2010). Optimum broadcasting in complete weighted-vertex graphs. In *36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)* (pp. 489–502).

Harutyunyan, H. A., & Kamali, S. (2017). Efficient broadcast trees for weighted vertices. *Discrete Applied Mathematics*, *216*, 598–608.

Harutyunyan, H. A., Laza, G., & Maraachlian, E. (2009, 05). Broadcasting in necklace graphs. In

ACM International Conference Proceeding Series (p. 253-256).

Harutyunyan, H. A., & Li, Z. (2017). Broadcast graphs using new dimensional broadcast schemes for Knödel graphs. In *Conference on Algorithms and Discrete Applied Mathematics (CALDAM)* (pp. 193–204).

Harutyunyan, H. A., & Li, Z. (2020). A new construction of broadcast graphs. *Discrete Applied Mathematics*, *280*, 144–155.

Harutyunyan, H. A., & Li, Z. (2021). The complexity of finding a broadcast center. In *International conference on algorithmic applications in management (aaim)* (pp. 57–70).

Harutyunyan, H. A., & Liestman, A. L. (1998). Messy broadcasting. *Parallel Processing Letters*, *8*(02), 149–159.

Harutyunyan, H. A., & Liestman, A. L. (1999). More broadcast graphs. *Discrete Applied Mathematics*, *98*(1-2), 81–102.

Harutyunyan, H. A., & Liestman, A. L. (2001a). Improved upper and lower bounds for k-broadcasting. *Networks*, *37*(2), 94–101.

Harutyunyan, H. A., & Liestman, A. L. (2001b). k-broadcasting in trees. *Networks*, *38*(3), 163–168.

Harutyunyan, H. A., & Liestman, A. L. (2003). On the monotonicity of the broadcast function. *Discrete Mathematics*, *262*(1-3), 149–157.

Harutyunyan, H. A., Liestman, A. L., Peters, J. G., & Richards, D. (2013). Broadcasting and gossiping. *Handbook of Graph Theory*, 1477–1494.

Harutyunyan, H. A., & Maraachlian, E. (2007). Linear algorithm for broadcasting in unicyclic graphs. In *International Computing and Combinatorics Conference (COCOON)* (pp. 372–382).

Harutyunyan, H. A., & Maraachlian, E. (2008). On broadcasting in unicyclic graphs. *Journal of Combinatorial Optimization*, *16*(3), 307–322.

Harutyunyan, H. A., & Maraachlian, E. (2009a). Broadcasting in Fully Connected Trees. In *15th IEEE International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 740–745).

Harutyunyan, H. A., & Maraachlian, E. (2009b). Linear Algorithm for Broadcasting in Networks

With No Intersecting Cycles. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)* (pp. 296–301).

Harutyunyan, H. A., & Shao, B. (2006). An efficient heuristic for broadcasting in networks. *Journal of Parallel and Distributed Computing*, *66*(1), 68–76.

Hedetniemi, S. M., & Hedetniemi, S. T. (1979). *Broadcasting by decomposing trees into paths of bounded length* (Tech. Rep.). University of Oregon: Technical Report CS-TR-79-16.

Hedetniemi, S. M., Hedetniemi, S. T., & Liestman, A. L. (1988). A survey of gossiping and broadcasting in communication networks. *Networks*, *18*(4), 319–349.

Hell, P., Klein, S., Nogueira, L. T., & Protti, F. (2004). Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics*, *141*(1-3), 185–194.

Hell, P., Klein, S., Protti, F., & Tito, L. (2001). On generalized split graphs. *Electronic Notes in Discrete Mathematics*, *7*, 98–101.

Hromkovič, J., Jeschke, C.-D., & Monien, B. (1993). Optimal algorithms for dissemination of information in some interconnection networks. *Algorithmica*, *10*(1), 24–40.

Hromkovič, J., Klasing, R., Monien, B., & Peine, R. (1996). Dissemination of information in interconnection networks (broadcasting & gossiping). In *Combinatorial Network Theory* (pp. 125–212). Springer.

Ivanova, M. (2019). *Optimization problems in communication networks and multi-agent path finding* (Unpublished doctoral dissertation). The University of Bergen.

Jansen, K., & Müller, H. (1995). The minimum broadcast time problem for several processor networks. *Theoretical Computer Science*, *147*(1-2), 69–85.

Kasim, H., March, V., Zhang, R., & See, S. (2008). Survey on parallel programming model. In *Network and parallel computing: Ifip international conference, npc 2008* (pp. 266–275).

Kenyon, R. W., Propp, J. G., & Wilson, D. B. (1999). Trees and matchings. *arXiv preprint math/9903025*.

Khachatrian, L., & Harutounian, O. (1990). Construction of new classes of minimal broadcast networks. In *Conference on Coding Theory, Armenia* (pp. 69–77).

Klasing, R., Monien, B., Peine, R., & Stöhr, E. A. (1994). Broadcasting in butterfly and DeBruijn networks. *Discrete Applied Mathematics*, *53*(1-3), 183–197.

Knödel, W. (1975). New gossips and telephones. *Discrete Mathematics*, *13*, 95.

Koh, J.-m., & Tcha, D.-w. (1991). Information dissemination in trees with nonuniform edge transmission times. *IEEE Transactions on Computers*, *40*(10), 1174–1177.

Kortsarz, G., & Peleg, D. (1995). Approximation algorithms for minimum-time broadcast. *SIAM Journal on Discrete Mathematics*, *8*(3), 401–427.

Labahn, R. (1994). A minimum broadcast graph on 63 vertices. *Discrete Applied Mathematics*, *53*(1-3), 247–250.

Li, C., Hart, T., Henry, K., & Neufeld, I. (2008, 12). Average-case "messy" broadcasting. *Journal of Interconnection Networks*, *9*, 487-505.

Liestman, A. L. (1985). Fault-tolerant broadcast graphs. *Networks*, *15*(2), 159–171.

Liestman, A. L., & Peters, J. G. (1988). Broadcast networks of bounded degree. *SIAM Journal on Discrete Mathematics*, *1*(4), 531–540.

Liestman, A. L., Richards, D., & Stacho, L. (2009). Broadcasting from multiple originators. *Discrete Applied Mathematics*, *157*(13), 2886–2891.

Lin, W., & Lam, P. C.-B. (2009). Star matching and distance two labelling. *Taiwanese Journal of Mathematics*, *13*(1), 211–224.

Maffray, F., & Preissmann, M. (1994). Linear recognition of pseudo-split graphs. *Discrete Applied Mathematics*, *52*(3), 307.

Maheo, M., & Saclé, J.-F. (1994). Some minimum broadcast graphs. *Discrete Applied Mathematics*, *53*(1-3), 275–285.

Maraachlian, E. (2010). *Optimal broadcasting in treelike graphs* (Unpublished doctoral dissertation). Concordia University.

Marathe, M. V., Ravi, R., Sundaram, R., Ravi, S., Rosenkrantz, D. J., & Hunt III, H. B. (1998). Bicriteria network design problems. *Journal of Algorithms*, *28*(1), 142–171.

Merris, R. (2003). Split graphs. *European Journal of Combinatorics*, *24*(4), 413–430.

Metcalfe, R. M., & Boggs, D. R. (1976). Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, *19*(7), 395–404.

Middendorf, M. (1993). Minimum broadcast time is NP-Complete for 3-regular planar graphs and deadline 2. *Information Processing Letters*, *46*(6), 281–287.

Mitchell, S., & Hedetniemi, S. (1980). A census of minimum broadcast graphs. *Journal of Combinatorics, Information, and System Sciences*, *5*, 141–151.

Müller, H. (1996). Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, *156*(1-3), 291–298.

Orlin, J. B. (2013). Max flows in O(nm) time, or better. In *Proceedings of the forty-fifth annual ACM Symposium on Theory of Computing (STOC)* (pp. 765–774).

Park, J.-H., & Chwa, K.-Y. (1994). Recursive circulant: A new topology for multicomputer networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)* (pp. 73–80).

Pelc, A. (1996). Fault-tolerant broadcasting and gossiping in communication networks. *Networks*, *28*(3), 143–156.

Proskurowski. (1981). Minimum broadcast trees. *IEEE Transactions on Computers*, *C-30*(5), 363-366.

Ravi, R. (1994). Rapid rumor ramification: Approximating the minimum broadcast time. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)* (pp. 202–213).

Richards, D., & Liestman, A. L. (1988). Generalizations of broadcasting and gossiping. *Networks*, *18*(2), 125–138.

Robledo, F., Rodríguez-Bocca, P., & Romero, P. (2020). Optimal broadcast strategy in homogeneous point-to-point networks. In *International Conference on Machine Learning, Optimization, and Data Science* (pp. 448–457).

Saclé, J.-F. (1996). Lower bounds for the size in four families of minimum broadcast graphs. *Discrete Mathematics*, *150*(1-3), 359–369.

Scheuermann, P., & Edelberg, M. (1981). Optimal broadcasting in point-to-point computer networks. *Dep. Elec. Eng. Comput. Sci., Northwestern Univ., Evanston, IL, Tech. Rep*.

Scheuermann, P., & Wu, G. (1984). Heuristic algorithms for broadcasting in point-to-point computer networks. *IEEE Computer Architecture Letters*, *33*(09), 804–811.

Schindelhauer, C. (2000). On the inapproximability of broadcasting time. In K. Jansen & S. Khuller

(Eds.), *Approximation algorithms for combinatorial optimization* (pp. 226–237). Berlin, Heidelberg: Springer Berlin Heidelberg.

Schrijver, A. (2002). On the history of the transportation and maximum flow problems. *Mathematical Programming*, *91*, 437–445.

Shao, B. (2006). On k-broadcasting in graphs. *Ph.D. Thesis, Concordia University*.

Slater, P. J., Cockayne, E. J., & Hedetniemi, S. T. (1981). Information dissemination in trees. *SIAM Journal on Computing*, *10*(4), 692–701.

Su, Y.-H., Lin, C.-C., & Lee, D. (2016). Broadcasting in weighted trees under the postal model. *Theoretical Computer Science*, *621*, 73–81.

Tamura, H., Tasaki, F., Sengoku, M., & Shinoda, S. (2005). Scheduling problems for a class of parallel distributed systems. In *2005 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 176–179).

Tyshkevich, R. I., & Chernyak, A. A. (1979). Canonical partition of a graph defined by the degrees of its vertices. *Isv. Akad. Nauk BSSR, Ser. Fiz.-Mat. Nauk*, *5*, 14–26.

Vandenberghe, L., & Andersen, M. S. (2015). Chordal graphs and semidefinite optimization. *Foundations and Trends in Optimization*, *1*(4), 241–433.

Xiao, J., & Wang, X. (1988). A research on minimum broadcast graphs. *Chinese Journal of Computers*, *11*, 99–105.

Zhang, R. Y., & Lavaei, J. (2018). Sparse semidefinite programs with near-linear time complexity. In *2018 IEEE Conference on Decision and Control (CDC)* (pp. 1624–1631).

Zheng, Y. (2019). *Chordal sparsity in control and optimization of large-scale systems* (Unpublished doctoral dissertation). University of Oxford.

Zhou, J.-g., & Zhang, K.-m. (2001). A minimum broadcast graph on 26 vertices. *Applied Mathematics Letters*, *14*(8), 1023–1026.

Čevnik, M., & Žerovnik, J. (2017). Broadcasting on cactus graphs. *Journal of Combinatorial Optimization*, *33*(1), 292-316.