Problems Related to Classical and Universal List Broadcasting

MohammadSaber GholamiNajarkola

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

November 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **MohammadSaber GholamiNajarkola**

Entitled: **Problems Related to Classical and Universal List Broadcasting**

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. John Xiupu Zhang

______________________ External Examiner
Dr. Franco Robledo

______________________ Examiner
Dr. Denis Pankratov

______________________ Examiner
Dr. Dhrubajyoti Goswami

______________________ Examiner
Dr. M. Reza Soleymani

______________________ Supervisor
Dr. Hovhannes A. Harutyunyan

Approved by ______________________
Lata Narayanan, Chair
Department of Computer Science and Software Engineering

November 9, 2022

____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Problems Related to Classical and Universal List Broadcasting

MohammadSaber GholamiNajarkola, Ph.D.

Concordia University, 2022

Broadcasting is a fundamental problem in the information dissemination area. In classical broadcasting, a message must be sent from one network member to all other members as rapidly as feasible. Although it has been demonstrated that this problem is NP-Hard for arbitrary graphs, it has several applications in various fields. As a result, the universal lists model, replicating real-world restrictions like the memory limits of nodes in large networks, is introduced as a branch of this problem in the literature. In the universal lists model, each node is equipped with a fixed list and has to follow the list regardless of the originator. In this study, we focus on both classical and universal lists broadcasting.

Classical broadcasting is solvable for a few families of networks, such as trees, unicyclic graphs, tree of cycles, and tree of cliques. In this study, we begin by presenting an optimal algorithm that finds the broadcast time of any vertex in a Fully Connected Tree (FCT_n) in $O(|V| \log \log n)$ time. An FCT_n is formed by attaching arbitrary trees to vertices of a complete graph of size n where $|V|$ is the total number of vertices in the graph. Then, we replace the complete graph with a Hypercube H_k and propose a new heuristic for the Hypercube of Trees (HT_k). Not only does this heuristic have the same approximation ratio as the best-known algorithm, but our numerical results also show its superiority in most experiments. Our heuristic is able to outperform the current upper bound in up to 90% of the situations, resulting in an average speedup of 30%. Most importantly, our results illustrate that it can maintain its performance even if the network size grows, making the proposed heuristic practically useful.

Afterward, we focus on broadcasting with universal lists, in which once a vertex is informed, it must follow its corresponding list, regardless of the originator and the neighbor from which it received the message. The problem of broadcasting with universal lists could be categorized into two sub-models: non-adaptive and adaptive. In the latter model, a sender will skip the vertices on its list from which it has received the message, while those vertices will not be skipped in the first model. In this study, we will present another sub-model called fully adaptive. Not only does this model benefit from a significantly better space complexity compared to the classical model, but, as will be proved, it is faster than the two other sub-models. Since the suggested model fits real-world network architectures, we will design optimal broadcast algorithms for well-known interconnection networks such as trees, grids, and cube-connected cycles. We also present an upper bound for tori under the same model. Then we focus on designing *broadcast graphs (bg)'s* under this model. A *bg* is a graph with minimum possible broadcast time from any originator. Additionally, a *minimum broadcast graph (mbg)* is a *bg* with the minimum possible number of edges. We propose mbg's on n vertices for $n \leq 10$ and sparse bg's for $11 \leq n \leq 14$ under the fully-adaptive model. Afterward, we introduce the first infinite families of *bg*'s under this model, and we prove that hypercubes are *mbg* under this model.

Later, we establish the optimal broadcast time of $k-$ary trees and binomial trees under the non-adaptive model and provide an upper bound for complete bipartite graphs. We also improved a general upper bound for trees under the same model. We then suggest several general upper bounds for the universal lists by comparing them with the messy broadcasting model.

Finally, we propose the first heuristic for this problem, namely `HUB-GA`: a Heuristic for Universal lists Broadcasting with Genetic Algorithm. We undertake various numerical experiments on frequently used interconnection networks in the literature, graphs with clique-like structures, and synthetic instances in order to cover many possibilities of industrial topologies. We also compare our results with state-of-the-art methods for classical broadcasting, which is proved to be the fastest model among all. Although the universal list model utilizes less memory than the classical model, our algorithm finds the same broadcast time as the classical model in diverse situations.

Acknowledgments

I wanna thank *me* for believing in me. I wanna thank *me* for doing all this hard work. I wanna thank *me* for having no days off. I wanna thank *me* for never quitting. I wanna thank *me* for always being a giver, and trying to give more than I receive. I wanna thank *me* for trying to do more right than wrong. I wanna thank *me* for just being *me* at all times!

Snoop Dog

Contents

List of Figures x

List of Tables xiii

1 Introduction 1

2 Literature Review and Preliminaries 4

2.1 Classical broadcasting 5

2.1.1 Minimum broadcast time problem 7

2.1.2 Broadcast graphs 9

2.2 Broadcasting with universal lists 11

2.3 Messy broadcasting 13

2.4 Comparison among broadcasting models 14

2.5 Commonly used topologies 14

2.5.1 Path P_n 14

2.5.2 Cycle C_n 15

2.5.3 Star S_n 15

2.5.4 Complete graph K_n 15

2.5.5 Complete bipartite graph $K_{m\times n}$ 16

2.5.6 Complete $k-$ary tree $T_{k,h}$ 16

2.5.7 Binomial tree T_d 17

2.5.8 Grid $G_{m\times n}$ 17

2.5.9 Tori $T_{m\times n}$ . . . 18
2.5.10 Hypercube H_d . . . 18
2.5.11 Cube-Connected Cycle CCC_d . . . 19
2.5.12 Shuffle Exchange SE_d . . . 19
2.5.13 DeBruijn DB_d . . . 20
2.5.14 Harary graph $H_{k,n}$ . . . 21

3 Optimal Broadcasting in Fully Connected Trees 23
3.1 Introduction . . . 23
3.2 A Broadcast Algorithm for Root Vertices . . . 25
3.2.1 Broadcast Algorithm . . . 26
3.2.2 Proof of Correctness . . . 32
3.2.3 Complexity Analysis . . . 38
3.3 A Broadcast Algorithm for Tree Vertices . . . 39
3.3.1 Broadcast Algorithm . . . 39
3.3.2 Proof of Correctness . . . 40
3.3.3 Complexity Analysis . . . 42

4 A Broadcasting Heuristic for Hypercube of Trees 43
4.1 Preliminaries, Motivation, and Literature Review . . . 43
4.2 The proposed heuristic . . . 45
4.3 Evaluation . . . 51
4.3.1 Experimental setup . . . 52
4.3.2 Numerical results . . . 53

5 Fully-adaptive Model for Broadcasting with Universal Lists 58
5.1 Fully-adaptive Model . . . 59
5.1.1 Assumptions, Architecture, and Applications . . . 62
5.2 Results on the Fully-adaptive Model . . . 64
5.2.1 Trees T . . . 64

5.2.2 Grids $G_{m \times n}$ 64

5.2.3 Tori $T_{m \times n}$ 65

5.2.4 Cube Connected Cycles CCC_d 66

5.2.5 Graphs with $B_{cl}(G) < B_{fa}(G)$ 69

6 Non-adaptive Broadcasting 71

6.1 Results on the non-adaptive model 71

6.1.1 Complete k-ary trees $T_{k,h}$ 71

6.1.2 Binomial Tree T_d 74

6.1.3 Complete Bipartite graph $K_{m \times n}$ 74

6.1.4 A general Upper bound for Trees 78

6.2 Comparison of universal list and messy broadcasting 80

7 Broadcast Graphs under the Fully-adaptive Model 83

7.1 Broadcast graphs under the fully-adaptive model 83

7.1.1 *mbg*'s for some values of $n \leq 16$ 84

7.1.2 General construction of bg's 86

7.2 Comparing broadcast time of various graphs 89

8 `HUB-GA`: A Heuristic for Universal lists Broadcasting using Genetic Algorithm 92

8.1 Motivation and an Example 93

8.1.1 An example 94

8.2 Related works on Genetic Algorithm 97

8.3 Methodology 100

8.3.1 Genes, Chromosomes, and Population 102

8.3.2 Fitness Function 103

8.3.3 Crossover 106

8.3.4 Mutation 107

8.3.5 Acceptance 108

8.3.6 Stopping criterion 109

8.3.7 Remarks 110

8.4 Experimental Results 111

8.4.1 Experiment 1 111

8.4.2 Experiment 2 115

8.4.3 Experiment 3 119

8.4.4 Experiment 4 126

9 Conclusion and Future work **132**

9.1 Conclusion 132

9.2 Future Work 134

9.3 Publications 135

Bibliography **137**

List of Figures

Figure 2.1 Path with $n = 5$ 14

Figure 2.2 Cycle with $n = 5$ 15

Figure 2.3 Star with $n = 6$ 15

Figure 2.4 Complete graph with $n = 5$ 16

Figure 2.5 Complete bipartite graph with $m = 4, n = 3$ 16

Figure 2.6 $k-$ary tree with $k = 4, h = 2$ 17

Figure 2.7 Binomial tree with $d = 4$ 17

Figure 2.8 Grid with $m = 4, n = 3$ 18

Figure 2.9 Torus with $m = 4, n = 3$ 18

Figure 2.10 Hypercube with $d = 3$ 19

Figure 2.11 Cube-Connected Cycle with $d = 3$ 20

Figure 2.12 Shuffle Exchange network with $d = 3$ 20

Figure 2.13 DeBruijn network with $d = 3$ 21

Figure 2.14 Harary graph with $k = 4, n = 12$ 21

Figure 2.15 Harary graph with $k = 5, n = 12$ 22

Figure 2.16 Harary graph with $k = 5, n = 13$ 22

Figure 3.1 A Fully Connected Tree FCT_n 24

Figure 3.2 A fully connected tree FCT_5 with 5 trees T_i rooted at i, $1 \leq i \leq 5$, and the originator vertex 1. The series of calls are shown with colorful arrows. 31

Figure 3.3 A Fully Connected Tree FCT_n, and a series of calls at time i under an arbitrary broadcast scheme S. 36

Figure 3.4 The position of vertex i and the sub-trees rooted at i. The originator, v, is located in T_{i_j}. We denote the rest of the sub-trees rooted at $i_m : m \neq j, 1 \leq m \leq d(i)$ as T'_i, in a way that $T_i = T_{i_j} \cup T'_i$. 40
Figure 4.1 HT_3, A hypercube of trees with dimension 3 45
Figure 4.2 Three Gaussian distribution curves 53
Figure 4.3 The performance of the algorithm when the value of μ or σ increases 56
Figure 5.1 a) A Graph G, b) Broadcast scheme of G under fully-adaptive model: $B^{\sigma}_{fa}(v_1, G) = 4$, c) Broadcast scheme of G under adaptive model: $B^{\sigma}_{a}(v_1, G) = 5$, d) Broadcast scheme of G under non-adaptive model: $B^{\sigma}_{na}(v_1, G) = 6$ 62
Figure 5.2 The relative location of an uninformed vertex u in an arbitrary CCC_d. . . . 68
Figure 5.3 a) A graph G with $B_{cl}(G) < B_{fa}(G)$, b) An example with $n = 6$. 69
Figure 6.1 $T_{k,h+1}$, An arbitrary $k-$ary tree with height $h + 1$. Observe that all nodes of this tree except for the leaves form a $k-$ary tree with height h. 72
Figure 7.1 a-c)*mbg*'s on 7,9, and 10 vertices under the fully-adaptive model. d-g) *bg*'s on 11,12,13 and 14 vertices under the fully-adaptive model. 85
Figure 7.2 Construction of graph G' 88
Figure 8.1 a) Graph G, b) Broadcast Scheme of G Under Fully Adaptive Model: $B^{\sigma}_{fa}(G, 3) = 3$, c) Broadcast Scheme of G Under Adaptive Model: $B^{\sigma}_{a}(G, 3) = 4$, d) Broadcast Scheme of G Under Non-Adaptive Model: $B^{\sigma}_{na}(G, 3) = 5$ 96
Figure 8.2 A schema of our methodology 104
Figure 8.3 The process of Selection, Crossover, and Mutation in `HUB-GA` 109
Figure 8.4 Karate club network (Zachary, 1977), used in Experiment 1 113
Figure 8.5 Results of Experiment 1: Impact of Population Size ($|p|$) 113
Figure 8.6 Results of Experiment 1: Impact of Stability variable (S_t) 114
Figure 8.7 Comparing the performance of `HUB-GA` with the known bounds on the broadcast time of a complete graph K_n of size $3 \leq n \leq 150$. 118
Figure 8.8 a) Ring of Clique $RC_{3,5}$, b) Windmill graph $W_{5,3}$ 119
Figure 8.9 Comparison of the performance of the `HUB-GA` and other heuristics in terms of Average broadcast time - Ring of cliques 124

Figure 8.10 Comparison of the performance of the HUB-GA and other heuristics in terms of Average broadcast time - Windmill graph 125

List of Tables

Table 3.1 The labels for the example graph after $t = 1$ 31

Table 3.2 The changes in the number of informed vertices following Lemma 3.2.2. . . 33

Table 3.3 Mapping between the notations of Theorem 4 of (Harutyunyan & Maraachlian, 2009b) and that of Algorithm 3 41

Table 4.1 Numerical Results for Ex1.1: $\mu = 3, \sigma = 1$ 54

Table 4.2 Numerical Results for Ex1.2: $\mu = 3, \sigma = 3$ 55

Table 4.3 Numerical Results for Ex1.3: $\mu = 5, \sigma = 5$ 55

Table 5.1 The Comparison Among Different Variants of Broadcasting 61

Table 5.2 An ordering of the vertices of the graph presented in Figure 5.1,a 63

Table 7.1 The known values of $B^{(fa)}(n)$ for $n < 15$ 86

Table 7.2 Broadcast time of well-known networks under models $M = \{cl, fa, a, na\}$. . 91

Table 8.1 An Ordering of Vertices for the Graph of Figure 8.1 96

Table 8.2 An optimal broadcast scheme for the Graph given in Figure 8.1 under fully-adaptive model 97

Table 8.3 An optimal broadcast scheme for the Graph given in Figure 8.1 under adaptive model 97

Table 8.4 An optimal broadcast scheme for the Graph given in Figure 8.1 under non-adaptive model 98

Table 8.5 Aims and scopes of the experiments 112

Table 8.6 Results of Experiment 2 116

Table 8.7 Results of Experiment 3 for $RC_{n,m}$, where $3 \leq n, m \leq 6$ 121

Table 8.8 Results of Experiment 3 for $W_{k,n}$, where $3 \leq k, n \leq 6$ 123

Table 8.9 Results of Experiment 4 for interconnection networks 129

Table 8.10 Results of Experiment 4 for connected complex networks 130

Notation Table

$G(V, E)$	An undirected connected 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
$B_{cl}(u, G)$	Classical broadcast time of vertex u in graph G
$B_{cl}(G)$	Classical broadcast time of graph G
$B^{(cl)}(n)$	Broadcast function for n vertices under classical model
$M_i, i \in \{1, 2, 3\}$	Messy broadcasting model i
$t_i(v), i \in \{1, 2, 3\}$	Messy broadcast time of vertex v under model i
$t_i(G), i \in \{1, 2, 3\}$	Messy broadcast time of graph G under model i
d_i	Degree of vertex i
$d(i)$	Number of subtrees of vertex i within T_i
l_v^u	List of vertex v when originator is vertex u
δ	Minimum degree of graph
Δ	Maximum degree of graph
na	Non-adaptive broadcasting with universal lists
a	Adaptive broadcasting with universal lists
fa	Fully-adaptive broadcasting with universal lists
$M \in \{na, a, fa\}$	Model of broadcasting with universal lists
l_v	Universal list of vertex v
$\Sigma_{(G)}$	All possible schemes for graph G
$\sigma_{n \times \Delta}$	A broadcast scheme with n rows and Δ columns
$B_M^\sigma(u, G)$	Broadcast time of vertex u following σ under M
$B_M^\sigma(G)$	Broadcast time of graph G following σ under M
$B_M(G)$	Broadcast time of graph G under M

$g^{(i)}$	Gene i, an ordering of neighbors of vertex i
$\|p\|$	Population size
$f_1(\sigma)$	Fitness function, maximum broadcast time
$f_2(\sigma)$	Fitness function, average broadcast time
K	Number of participants of a $K-$way tournament
c_p	Crossover point
p_1	A chromosome: parent number 1
c_1	A chromosome: offspring number 1
$g^{(i)}(p_j)$	The i^{th} gene of parent j
S_t	Stability variable

Chapter 1

Introduction

High-performance interconnection networks connect compute nodes on a cluster computer so that they can communicate with each other. A cluster computer comprises multiple compute nodes, each having several processing elements. The design of such networks is decisive, specifically in High-Performance Computing (HPC) systems (Lee, Hong, & Li, 2021; Rocher-Gonzalez, Escudero-Sahuquillo, García, & Quiles, 2017). Moreover, an interconnection network is utilized whenever synchronization among the processing elements or exchange of intermediate results is required. More importantly, the interconnection network topology and the routing scheme determine the overall network performance and the entire system (Al Faisal, Rahman, & Inoguchi, 2017). In particular, low latency, high bandwidth, and memory-efficient communication between the compute nodes are necessary for HPC applications to scale appropriately on multiple machines. Therefore, studying interconnection networks and, specifically, the algorithms that move the data around the processing units are pivotal in the performance of such applications (Rocher-Gonzalez et al., 2017), particularly with limited memory.

A vital problem in information dissemination is *broadcasting*, which refers to distributing a piece of information in a communication network. In particular, a message held initially by a single network member must be promptly transmitted to all network members. This is achieved by placing a series of calls over the network's communication links while respecting the following limitations: A call involves precisely two network members and is executed in a single unit of time. Although a node can only pass the message to one of its neighbors at a time, a vertex may receive the message from multiple senders at each time unit. The applications of this problem include, but are not limited to, local computer networks (Metcalfe & Boggs, 1976), malware diffusion (Karyotis & Khouzani, 2016), multiprocessor systems (Chen, Shin, & Kandlur, 1990), and parallelism (Varvarigos & Bertsekas, 1995). Besides, the time it takes to transmit a message between two communication sites or the message delay is one factor affecting a network's performance (Hasson & Sipper, 2004).

A network is modeled with an undirected graph $G = (V, E)$, where V is a set of vertices representing the network members, and E is a set of bidirectional communication links between the members of the network. It is generally assumed that G is a connected graph. The broadcast time of a vertex $v \in V(G)$ under the classical model is denoted by $B_{cl}(v, G)$ and is defined to be the minimum number of time units required for a message to be transmitted to all members of V, originating from v. By definition, the broadcast time of the graph G under the classical model is the maximum broadcast time of any vertex v in G, and it is denoted by $B_{cl}(G)$: $B_{cl}(G) = \max\{B_{cl}(v, G) | v \in V(G)\}$. Not only is it NP-Hard to find $B_{cl}(G)$ and $B_{cl}(v, G)$ for arbitrary graphs and originators (Garey & Johnson, 1983), it is proved that this problem remains NP-Hard in more restricted families of networks (Dinneen, 1994; Jakoby, Reischuk, & Schindelhauer, 1998).

Achieving optimal broadcasting in a network requires deep knowledge of network topology. To illustrate, not only do network members need to be aware of their neighbors' states, but they should also know the origin of the message, which is memory-inefficient. Hence, several settings of this problem have been defined in the literature to simulate real-world networks, i.e., messy broadcasting and the universal list model. In the first model, when a node is informed, it randomly selects one of its neighbors and sends the message to that node (Ahlswede, Haroutunian, & Khachatrian, 1994; Harutyunyan & Liestman, 1998). In the latter model, network vertices are equipped with a universal list of their neighbors. When a vertex receives the message, it follows its list and passes the message to the vertices of its list (Diks & Pelc, 1996). Nevertheless, in both models, a member needs local knowledge of their neighbors.

In this study, we focus on both classical and universal lists broadcasting. The contributions of this manuscript are as follows:

- In Chapter 3, we present an algorithm for classical broadcasting in Fully-Connected Trees (FCT_n), which are constructed by attaching arbitrary trees to a complete graph of size n. We also prove that the algorithm is optimal.

- In Chapter 4, we replace the complete graph with a Hypercube of dimension k and propose a novel heuristic for classical broadcasting in a Hypercube of Trees HT_k.

- Starting from Chapter 5, we study the universal lists broadcasting. We first propose a new sub-model for broadcasting with universal lists while giving some insight into its applications in real-world networks. We then provide the exact broadcast time of Trees, Grids, and Cube-Connected Cycles under our newly introduced fully adaptive model. We also present an additive approximation algorithm for Tori.

- In Chapter 6, we study non-adaptive broadcasting with universal lists by proposing the exact value of broadcast time for complete $k-$ary trees and binomial trees. Then we provide an upper bound for complete bipartite graphs and improve the general upper bounds for trees.

- In Chapter 7, we study the problem of broadcast graphs under the fully-adaptive model, which are the cheapest networks in which broadcasting could be finished as soon as possible. To this aim, we give the exact value of broadcast functions for $3 \leq n \leq 10$, upper bounds for $11 \leq n \leq 14$, and some general upper bounds on broadcast function by giving a constructive method. We introduce Hypercube as the first infinite family of minimum broadcast graphs under the fully adaptive model.

- Lastly, in Chapter 8, we propose the first heuristic for universal lists broadcasting. Out heuristic is based on the well-known Genetic Algorithm, and we study its performance in various networks by conducting several experiments.

Chapter 2

Literature Review and Preliminaries

Broadcasting is a problem in which a sender, usually called the $originator$, has a piece of information in a network and wishes to inform all network members of this message. This is achieved by placing a series of calls over the network's communications links while respecting the following conditions (Hedetniemi, Hedetniemi, & Liestman, 1988):

(1) Each call involves exactly two members,

(2) Each call needs precisely one unit of time,

(3) A vertex can participate in only one call in each unit of time,

(4) And a vertex can only make a call toward its adjacent vertices.

The process ends when all members are informed. This problem could be categorized into different settings if being looked at from various perspectives:

- **Directions of the edges:** telegraph communication model vs. telephone communication model.

- **Number of neighboring processors:** 1-port communication model (Fraigniaud & Lazard, 1994), k-port communication model (Harutyunyan & Liestman, 2001b), link bound model (Fraigniaud & Lazard, 1994).

- **Time taken to send a message between two nodes:** constant model vs. linear model (Beauquier, Perennes, & Delmas, 2001).
- **Communication setup with neighbors:** Vertex disjoint path (Farley, 2004), Edge disjoint path (Farley, 2004), (i, j) mode (Even & Monien, 1989), Radio broadcasting (Alon, Bar-Noy, Linial, & Peleg, 1991), Universal lists broadcasting (Diks & Pelc, 1996), Messy broadcasting (Ahlswede et al., 1994), Multiple message broadcasting (Bar-Noy & Kipnis, 1994), Fault-tolerant broadcasting (Ahlswede, Gargano, Haroutunian, & Khachatrian, 1996).

In this study, we follow the definitions of the 1-port, constant time model (Fraigniaud & Lazard, 1994), in which a node can communicate with 1 neighbor at a time, and the transmitting time is constant. Robledo et al. argued that one-to-one communication is experimentally faster than a setup in which a sender may update all its neighbors simultaneously (Robledo, Rodríguez-Bocca, & Romero, 2020). The main focus of this study is on the classical model of broadcasting, universal lists broadcasting, and messy broadcasting, which are discussed subsequently.

2.1 Classical broadcasting

For a vertex $v \in V(G)$, we denote by $B_{cl}(v, G)$ the classical broadcast time of that vertex in the graph, which is the minimum time steps required to finish the broadcast process, originating from v. The broadcast time of the graph, $B_{cl}(G)$, is the maximum broadcast time of all possible originators:

$$B_{cl}(G) = \max_{v \in V(G)} \{B_{cl}(v, G)\} \tag{1}$$

The problem of finding $B_{cl}(v, G)$ and $B_{cl}(G)$ are both NP-Hard for arbitrary graphs and originators (Garey & Johnson, 1983; Slater, Cockayne, & Hedetniemi, 1981). Besides, the problem remains NP-Hard in more restricted families such as bounded degree graphs (Dinneen, 1994) and regular planar graphs (Jakoby et al., 1998; Middendorf, 1993). The formal definition of the decision version of this problem is 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 $E_i \subseteq E$, $V_K = V$, 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) $V_i = V_{i-1} \cup \{v : \{u, v\} \in E_i\}$?

When $|V_0| = 1$, it is the case when broadcasting starts from a single originator, which remains NP-Complete (Garey & Johnson, 1983). The NP-Completeness proof of the decision version is presented in (Slater et al., 1981), and a reduction from the three-dimension matching (3DM) problem has been utilized for the proof. Also, this problem cannot be approximated in polynomial time for an arbitrary graph within a ratio of $3 - \epsilon$ for any $\epsilon > 0$ unless $P = NP$ (Elkin & Kortsarz, 2005).

The broadcast scheme[1] of the classical model is the series of calls placed in the network. This could be interpreted as an ordering of the neighbors of each vertex (Diks & Pelc, 1996). Consider vertex u as the originator. Once a vertex v receives the source message, it will utilize its list, denoted by l_v^u, and pass the information to its uninformed neighbors following the order of its list. A valid broadcast scheme contains the lists so that every network member is equipped with the message after placing all calls. However, the vital issue with the classical model is that those lists differ for various originators. Therefore, each vertex must maintain several lists depending on the originator and adapt its behavior accordingly. In particular, a vertex v has to maintain up to $|V|$ different lists, denoted by $l_v^u, \forall u \in V(G)$. Two principal problems are defined in this area:

- **Broadcast time problem:** in which the goal is to find the broadcast time of a vertex or a graph.
- **Network design problem:** where the goal is to design a network in which broadcasting could be finished as quickly as possible from any originator.

These problems are discussed in 2.1.1 and 2.1.2, respectively.

[1] or broadcast algorithm

2.1.1 Minimum broadcast time problem

As mentioned before, the MBT problem is a classical NP-Complete problem. Therefore, three directions are followed by researchers to tackle this problem:

(1) **Exact algorithms** solve the problem optimally using different techniques such as linear programming or solve it optimally for a particular family of graphs in a reasonable amount of time.

(2) **Approximation algorithms** provide a worst-case performance guarantee in both computational time, and solution quality,

(3) **Heuristics** typically focus on the average empirical behavior of the algorithms.

In the rest of this section, some studies related to each of these are discussed. Also, since there are several papers dealing with finding the broadcast time of graphs, we mention a few survey papers from which the reader can trace back all the previous works (Fraigniaud & Lazard, 1994; Harutyunyan, Liestman, Peters, & Richards, 2013; Hedetniemi et al., 1988; Hromkovič, Klasing, Monien, & Peine, 1996).

Exact algorithms

Firstly, some researchers designed exact approaches by formulating an optimization version of this problem. This includes a Dynamic Programming algorithm suggested in (Scheuermann & Wu, 1984) and the ILP models proposed in (de Sousa et al., 2018) and (Ivanova, 2019). Currently, the ILP method offered in (de Sousa et al., 2018) is believed to be the best exact method. These methods can only effectively solve this problem for networks with up to 50 nodes in a reasonable amount of time, making them unsuitable for use in real-world networks.

Secondly, some researchers tried to solve the problem optimally for a particular family of networks. Slater et al. (Slater et al., 1981) made the first step in this category and offered a linear algorithm for trees. Other researches include grid and tori (Farley & Hedetniemi, 1978), Cube Connected Cycle (Liestman & Peters, 1988), and Shuffle Exchange (Hromkovič, Jeschke, & Monien, 1993). Later on, more algorithms for non-trivial topologies such as unicyclic graphs (Harutyunyan

& Maraachlian, 2007, 2008) and Tree of Cycles (Harutyunyan & Maraachlian, 2009b) were developed, to name a few. Also, there is an $O(n^{4k+5})$ broadcast algorithm for partial k-trees (Dessmark, Lingas, Olsson, & Yamamoto, 1998).

Several papers have studied the exact broadcast time of a specific class of graphs or investigated the lower and upper bounds of the broadcast time (Ahlswede et al., 1996, 1994; Averbuch, Peeri, & Roditty, 2017; Comellas, Harutyunyan, & Liestman, 2003; Comellas & Hell, 2003; Farley, 2004; Harutyunyan, 2000, 2006; Harutyunyan & Liestman, 2001a, 2001b; Harutyunyan & Maraachlian, 2009b).

Approximation algorithms

Various studies provide approximation algorithms for specifying the broadcast time of a particular vertex in an arbitrary network. One of the first research in this area gives an additive $(\sqrt{|V|})$-approximation algorithm for finding the broadcast time of any graph (Kortsarz & Peleg, 1995). These results were improved by Ravi, (Ravi, 1994) who proposed a $(\frac{\log^2 |V|}{\log\log |V|})$-approximation algorithm.

Besides, Bar-Noy et al. (Bar-Noy, Guha, Naor, & Schieber, 1998) worked on the multicasting problem in which the aim is to inform a subset T of vertices. They gave a $O(\log |T|)$-approximation for multicasting which could be trivially generalized to a $(\log |V|)$-approximation algorithm for broadcasting. However, they used linear programming, which is computationally expensive. Elkin et al. (Elkin & Kortsarz, 2005) introduced an algorithm with a $(\log |V|)$ approximation ratio to overcome this drawback. These bounds were soon improved by themselves in (Elkin & Kortsarz, 2006) in which a $(\frac{\log |T|}{\log\log |T|})$-approximation algorithm for multicasting (or a $(\frac{\log |V|}{\log\log |V|})$-approximation solution for broadcasting) is proposed. This algorithm is the best approximation known for this problem.

Various papers also designed approximation algorithms for a specific family of networks, such as k-path graphs (Bhabak & Harutyunyan, 2019), k-cycle graphs (Bhabak & Harutyunyan, 2015), graphs with known broadcast time of the base graph (Bhabak & Harutyunyan, 2022), and Harary graphs and graphs similar to Harary graphs (Bhabak, Harutyunyan, & Kropf, 2017; Bhabak, Harutyunyan, & Tanna, 2014), to name a few.

Heuristics

The authors of (Elkin & Kortsarz, 2005) proposed a heuristic algorithm considering directed graphs. In (Beier & Sibeyn, 2000) two heuristics, called `matching heuristic` and `coloring heuristic`, are presented. They also present the `Round-Heuristic`, which has a running time of $O(Rnm \log n)$ where R is the number of rounds taken for broadcasting, n is the number of vertices, and m is the number of edges in the graph.

The `Tree-based` approach is presented in (Harutyunyan & Shao, 2006), which reduces the complexity of each round to $O(m)$. In (Harutyunyan & Wang, 2010), the `NTBA` is introduced that applies Random Heuristic and Semi-Random Heuristic algorithms which decrease the total time complexity to $O(m)$. Their results show that `NTBA` performs better than the previous approaches in the real-world network models.

We refer the reader to (Albin, Kottegoda, & Poggi-Corradini, 2020; Fraigniaud & Vial, 1997, 1999; Harutyunyan & Jimborean, 2014; Ravi, 1994; Scheuermann & Wu, 1984) for more results in this category.

2.1.2 Broadcast graphs

A graph G on n vertices is called a *broadcast graph (bg)* if $B_{cl}(G) = \lceil \log n \rceil$. A *bg* with the minimum number of edges is called a *minimum broadcast graph (mbg)*. The minimum number of edges is called the *broadcast function* and denoted by $B^{(cl)}(n)$. Studying *mbg*'s, as well as lower/upper bounds on the value of the *broadcast function*, has attracted a lot of attention from researchers since an *mbg* represents the cheapest graph (in terms of the number of edges), where broadcasting can be accomplished in the minimum possible time.

This problem was proposed in (Farley, Hedetniemi, Mitchell, & Proskurowski, 1979), where the authors also determined the value of $B^{(cl)}(n)$ for $n \leq 15$ and $n = 2^k$ while introducing hypercubes as the first infinite family of *mbg*'s. Later, Knödel graph (Knödel, 1975) was proved to be the second infinite family of *mbg*'s for $n = 2^k - 2$ (Dinneen, Fellows, & Faber, 1991; Khachatrian & Harutounian, 1990). Finally, Park and Chwa proposed recursive circulant graphs as a non-isomorphic alternative to hypercubes for *mbg*'s on $n = 2^k$ vertices (Park & Chwa, 1994).

Except for the three mentioned families, there is no other infinite family of *mbg*'s. The value of $B^{(cl)}(n)$ is also known for $n = 17$ (Mitchell & Hedetniemi, 1980), $n = 18, 19$ (Bermond, Hell, Liestman, & Peters, 1992; Xiao & Wang, 1988), $n = 20, 21, 22$ (Maheo & Saclé, 1994), $n = 26$ (Saclé, 1996; Zhou & Zhang, 2001), $n = 27, 28, 29$ (Saclé, 1996), $n = 30, 31$ (Bermond et al., 1992), $n = 58, 61$ (Saclé, 1996), $n = 63$ (Labahn, 1994), $n = 127$ (Harutyunyan, 2008), and $n = 1023, 4095$ (Shao, 2006). Although the last four mentioned values of n are all in the form of $n = 2^k - 1$, there is no infinite family of *mbg*'s for the general case. Also, note that the value of $B^{(cl)}(n)$ is not known for some small values of n such as 23, 24, and 25.

Due to the difficulty of this problem, there are two general directions to follow:

- **Upper bounds on broadcast function:** This is achieved by constructing sparse *bg*'s. For example, in (Farley, 1979), two or three *bg*'s are combined to create a bigger *bg* which shows an upper bound of $B^{(cl)}(n) \leq \frac{n}{2}\lceil \log n \rceil$. Later, Chau and Liestman generalized this idea by using up to 7 smaller *bg*'s (Chau & Liestman, 1985). A tight asymptotic bound of $B^{(cl)}(n) \in \Theta(L(n).n)$ is given in (Grigni & Peleg, 1991), where they proved that $\frac{L(n)-1}{2}n \leq B^{(cl)}(n) \leq (L(n)+2)n$, in which $L(n)$ is the number of consecutive leading 1's in the binary representation of $n-1$. Furthermore, the compounding method utilizes the vertex cover of the graph (Khachatrian & Harutounian, 1990) or the solid vertex cover of the graph (Bermond, Fraigniaud, & Peters, 1995). More recently, compounding binomial trees with hypercubes improved the upper bound on $B^{(cl)}(n)$ for many values of n (Averbuch, Shabtai, & Roditty, 2014; Harutyunyan & Li, 2017).

- **Lower bounds on broadcast function:** proposing lower bounds on the broadcast function is more challenging since most lower bound proofs are based only on vertex degree. In (Gargano & Vaccaro, 1989), a lower bound of $B^{(cl)}(n) \geq \frac{n}{2}(\lfloor \log n \rfloor - \log(1+2^{\lceil \log n \rceil} - n))$ is suggested for any n. Another lower bound of $B^{(cl)}(n) \geq \frac{n}{2}(m-p-1)$ is provided in (König & Lazard, 1994), where m is the length of the binary representation of n and p is the index of the leftmost 0 bit. Additionally, some lower bounds for special values of n are proposed in the literature, such as $B^{(cl)}(n) \geq \frac{m^2(2^m-1)}{2(m+1)}$ for $n = 2^m - 1$ (Labahn, 1994). Also, some other lower bounds on $B^{(cl)}(2^m-3)$, $B^{(cl)}(2^m-4)$, $B^{(cl)}(2^m-5)$, and $B^{(cl)}(2^m-6)$ are given in (Saclé, 1996).

To the best of our knowledge, this problem has only been studied for classical broadcasting and the $k-$broadcasting model (Harutyunyan & Liestman, 2001a).

2.2 Broadcasting with universal lists

In classical broadcasting, a *call* is only initiated from an informed vertex to an uninformed vertex, which requires comprehensive knowledge of the whole network for every vertex. In other words, once informed, a vertex sends the message to some of its uninformed neighbors in a particular order. This ordering, however, is different for each originator, as mentioned before. Thus, each vertex has to know the originator to choose the corresponding list and adapt its behavior. It also should maintain a significantly extensive list according to different originators. This is inefficient in real-world networks due to the increased message bits and the need for larger local memory (Diks & Pelc, 1996; J.-H. Kim & Chwa, 2005).

To handle the above-mentioned drawbacks, another variant of broadcasting is introduced (Diks & Pelc, 1996; Rosenthal & Scheuermann, 1987). Suppose every network member is given a universal list, and it has to follow the list regardless of the originator. So, when a vertex v receives the message, it should transmit the message to its neighbors following the ordering given in its list l_v. In this model, we drop the superscript in the notation of l_v as the list is universal for all originators. There are two sub-models defined using universal lists:

(1) *Non-adaptive:* Once a vertex v receives the message, it will re-transmit it to all the vertices on its list, even if v has received it from some vertices of its list. The broadcast time of a graph G following this model is denoted by $B_{na}(G)$.

(2) *Adaptive:* Once informed, a vertex v will send the message to its neighbors according to its list, but it will skip the neighbors from which it has received the message. The broadcast time of a graph G following this model is denoted by $B_a(G)$.

Considering graph $G = (V, E)$ in which $|V| = n$, a broadcast scheme for non-adaptive or adaptive models can be viewed as a matrix $\sigma_{n\times\Delta}$, where row i of σ corresponds to an ordering of the neighbors of vertex v_i. Assuming this vertex has degree d_i, the cells $\sigma_{[i][d_i+1]}, \sigma_{[i][d_i+2]}, \cdots, \sigma_{[i][\Delta]}$ will be NULL. By definition: $\Delta = \max\{d_i : 1 \leq i \leq n\}$. We denote all possible schemes for a graph G by $\Sigma_{(G)}$. When it is clear from the context, we may omit the subscript (G).

Let M be one of the two models using universal lists ($M \in \{na, a\}$) and fix a graph G. For any broadcast scheme $\sigma \in \Sigma$, we denote by $B_M^{\sigma}(v, G)$ the time steps needed to inform all the vertices in G from the source v while following the scheme σ under model M. Moreover, the broadcast time of a graph G under model M with scheme σ, $B_M^{\sigma}(G)$, is defined as the maximum $B_M^{\sigma}(v, G)$ over all possible originators $v \in V(G)$. Lastly, $B_M(G)$ is the minimum $B_M^{\sigma}(G)$ over all possible schemes $\sigma \in \Sigma$:

$$\begin{aligned} B_M^{\sigma}(G) &= \max_{v \in V(G)} \{B_M^{\sigma}(v, G)\} \\ B_M(G) &= \min_{\sigma \in \Sigma} \{B_M^{\sigma}(G)\} \end{aligned} \tag{2}$$

This problem was first discussed indirectly by Slater et al. (Slater et al., 1981). They proved that for any tree T, $B_{cl}(T) = B_a(T)$. However, the first formal definition of the problem of broadcasting with universal lists was given in (Rosenthal & Scheuermann, 1987), where the authors suggested an algorithm for constructing optimal broadcast schemes for trees under the adaptive model. Afterward, Diks and Pelc (Diks & Pelc, 1996) distinguished between non-adaptive and adaptive models and designed optimal broadcast schemes for cycles and grids under both models. They also gave upper bounds for tori and complete graphs for adaptive and non-adaptive models. The upper bounds for tori were improved in (Harutyunyan & Taslakian, 2004).

Later, Kim and Chwa (J.-H. Kim & Chwa, 2005) designed non-adaptive broadcast schemes for paths and grids. They also developed upper bounds for complete graphs and hypercubes under non-adaptive models. Finally, the lower and upper bounds for trees under the non-adaptive model have tightened in (Harutyunyan, Liestman, Makino, & Shermer, 2011) as well as the upper bounds on general graphs. The authors of (Harutyunyan et al., 2011) also proposed a polynomial-time algorithm for finding $B_{na}(T)$ for any tree T.

2.3 Messy broadcasting

This model, introduced in (Ahlswede et al., 1994), is similar to the universal list model, but the network nodes have even more limited memory. In messy broadcasting, every informed vertex randomly chooses a neighbor and sends the message. Thus, the goal is to study the worst behavior of the network members. There are three sub-models defined for messy broadcasting:

(1) *Model M_1:* Each vertex knows the state of its neighbors, informed or uninformed. Therefore, once a vertex gets informed, it only has to send the message to its uninformed neighbors in some arbitrary order.

(2) *Model M_2:* Each vertex knows from which vertices it has received the message and considers those as informed vertices. Thus, once vertex v gets informed, it will randomly send the message to the neighbors who have not sent it to v before.

(3) *Model M_3:* Each vertex only knows to which vertices it has sent the message and will consider them as informed neighbors; all other neighbors are regarded as uninformed for this vertex. Therefore, a vertex will send the message to all of its neighbors in an arbitrary order once it gets informed.

The broadcast time of a vertex v under model M_i is denoted by $t_i(v)$, for $i = 1, 2, 3$, and it is defined as the *maximum* number of time units required to complete broadcasting originating from vertex v over all possible broadcast schemes. Also, the broadcast time of the graph G under model M_i is denoted by $t_i(G)$, for $i = 1, 2, 3$, and is the maximum broadcast time of any vertex v of G.

The exact value of $t_i(G)$ for complete graphs, paths, cycles, and complete d-ary trees are known for $i = 1, 2, 3$ (Harutyunyan & Liestman, 1998). Also, in (Comellas et al., 2003), multidimensional directed tori and complete bipartite graphs are studied. Moreover, the average-case messy broadcasting time of various networks such as stars, paths, cycles, complete d-ary trees, and hypercubes are studied in (C. Li, Hart, Henry, & Neufeld, 2008).

2.4 Comparison among broadcasting models

In summary, the differences among these three broadcasting models are as follows: In the classical model, being the quickest among all, there exists an omniscient who knows the network's exact situation at every single unit of time. Therefore, it will guide network members to broadcast as efficiently as possible while adapting their behavior according to the originator. This omniscient may be considered as a network manager who is provided with sufficient memory for each node and allows them to change their behavior depending on the originator.

In the universal list model, however, the network manager cannot behave as prodigal as in the previous model since the memory is limited regarding each member. Therefore, the behavior of all members, a.k.a, the universal lists, are to be set beforehand, and then, the broadcasting will be performed according to the lists. Indeed, the network manager tries to minimize the broadcast time of the network as much as possible.

In contrast, in messy broadcasting, no one monitors the network situation, and the members will act randomly. Hence, the worst behavior of the network with respect to the broadcast time is of interest.

2.5 Commonly used topologies

This section presents the details of some network topologies used throughout this study.

2.5.1 Path P_n

A path P_n is a graph on n vertices, $V = \{1, \cdots, n\}$, and the following set of edges: $E = \{(i, i+1) | 1 \leq i < n\}$. P_n has $n-1$ edges, diameter of $n-1$ and a maximum degree of 2. Also, $B_{cl}(P_n) = n-1$. Figure 2.1 portrays P_5.

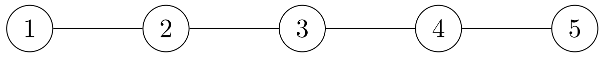


Figure 2.1: Path with $n = 5$

2.5.2 Cycle C_n

A cycle C_n is a graph on n vertices, $V = \{1, \cdots, n\}$, and the following set of edges: $E = \{(i, i+1) | 1 \leq i < n\} \cup \{(1, n)\}$. C_n has n edges, diameter of $\lfloor \frac{n}{2} \rfloor$ and a maximum degree of 2. Also, $B_{cl}(C_n) = \lceil \frac{n}{2} \rceil$. Figure 2.2 portrays C_5.

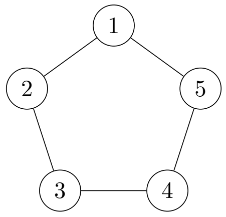


Figure 2.2: Cycle with $n = 5$

2.5.3 Star S_n

A star S_n is a graph on n vertices, $V = \{1, \cdots, n\}$, and the following set of edges: $E = \{(i, n) | 1 \leq i < n\}$. S_n has $n-1$ edges, a diameter of 2, and a maximum degree of $n-1$. Also, $B_{cl}(S_n) = n - 1$. Figure 2.3 portrays S_6.

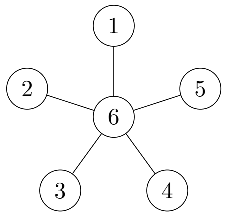


Figure 2.3: Star with $n = 6$

2.5.4 Complete graph K_n

A complete graph, a.k.a clique, K_n is a graph on n vertices, $V = \{1, \cdots, n\}$, and all possible edges. K_n has $\frac{n(n-1)}{2}$ edges, a diameter of 1, and a maximum degree of $n-1$. Also, $B_{cl}(K_n) = \lceil \log n \rceil$. Figure 2.4 portrays K_5.

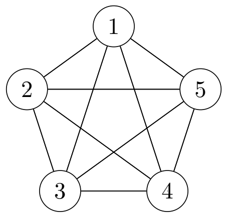


Figure 2.4: Complete graph with $n = 5$

2.5.5 Complete bipartite graph $K_{m\times n}$

A complete bipartite graph consists of two separate partitions; partition M with m nodes and partition N with n nodes. A vertex in a partition is connected to all vertices in the other partition but is not connected to any other vertex in the same partition. Subsequently, the degree of a node in partition M equals n, whereas a node in partition N has exactly m neighbors. A complete bipartite graph consists of $m + n$ vertices and $m \times n$ edges. Figure 2.5 shows $K_{4\times 3}$

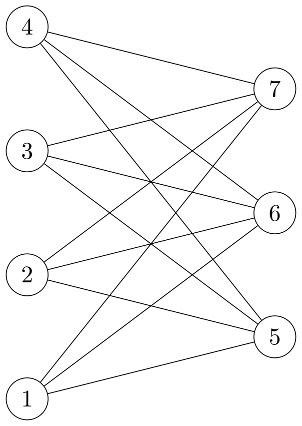


Figure 2.5: Complete bipartite graph with $m = 4, n = 3$

2.5.6 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 degree of other internal vertices is $k + 1$, and the degree of a leaf is 1. A complete k-ary tree, $T_{k,h}$, is a rooted k-ary tree in which all leaves are at the same level h. $T_{k,h}$ has $\frac{k^{h+1}-1}{k-1}$ vertices, diameter of $2h$, and maximum degree of $k + 1$. Figure 2.6 portrays $T_{4,2}$.

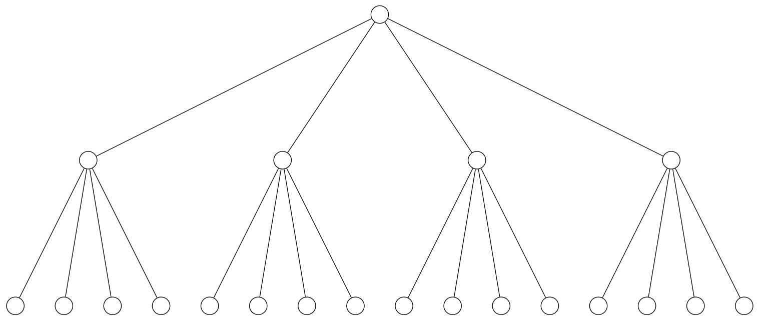


Figure 2.6: $k-$ary tree with $k = 4, h = 2$

2.5.7 Binomial tree T_d

A binomial tree of dimension d, T_d, has a recursive definition. T_0 is the root of a tree with one node, and T_d consists of two copies of T_{d-1} in which an edge connects the root vertices. One of the root vertices is randomly chosen as the new root of T_d. A T_d has 2^d vertices, a diameter of $2d - 1$, and a maximal degree of d. Figure 2.7 shows T_4, which consists of two copies of T_3.

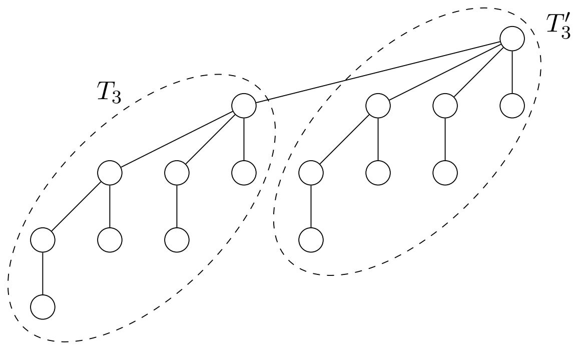


Figure 2.7: Binomial tree with $d = 4$

2.5.8 Grid $G_{m\times n}$

A two-dimensional Grid, $G_{m\times n}$, is a planar graph whose vertices are assigned integer coordinates (x, y). Two vertices are neighbors if they agree in one coordinate and differ by one in the other. $G_{m\times n}$ has m columns and n rows, i.e. $1 \le x \le m$ and $1 \le y \le n$. It is proved that $B_{cl}(G_{m\times n}) = m + n - 2$ (Farley & Hedetniemi, 1978). The diameter of $G_{m\times n}$ is also equal to $m + n - 2$. Figure 2.8 shows $G_{4\times 3}$.

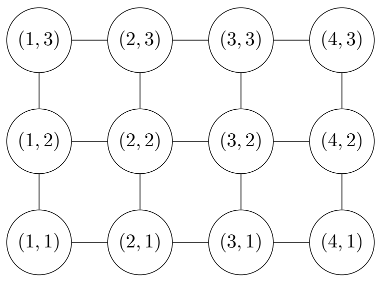


Figure 2.8: Grid with $m = 4, n = 3$

2.5.9 Tori $T_{m\times n}$

A torus $T_{m\times n}$ is a grid with m columns and n rows equipped with wraparound edges. For instance, a vertex on the first column and ith row is also connected to the vertex on the last column and ith row. Also, $B_{cl}(T_{m\times n}) = \lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor$, if m and n are even, and $\lfloor \frac{m}{2} \rfloor + \lfloor \frac{n}{2} \rfloor + 1$ otherwise (Farley & Hedetniemi, 1978). Figure 2.9 portrays $T_{4\times 3}$.

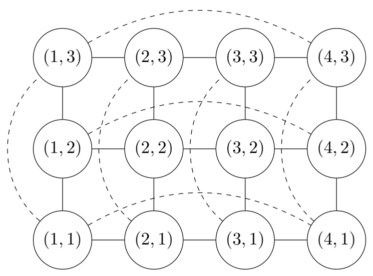


Figure 2.9: Torus with $m = 4, n = 3$

2.5.10 Hypercube H_d

The Hypercube H_d of dimension d has the following set of vertices: $V_{H_d} = \{0, 1\}^d$, in which $\{0, 1\}^d$ denotes the set of binary strings with length d. Therefore, a vertex $v \in V_{H_d}$ could be represented by $\alpha = a_0a_1 \cdots a_{d-1}$, where $a_i \in \{0, 1\}$, $0 \le i \le d - 1$. The edges of H_d are denoted by E_{H_d} and it is formed as follows: A vertex $\alpha = a_0a_1 \cdots a_{d-1} \in \{0, 1\}^d$ is connected to $\alpha(i)$ for each $i \in \{0, 1, \cdots, d - 1\}$, where $\alpha(i) = a_0a_1 \cdots a_{i-1}\bar{a}_ia_{i+1} \cdots a_{d-1}$. Note that $\bar{a}$ represents the binary complement of a. We call $\alpha(i)$ the ith dimensional neighbour of α. H_d has 2^d vertices, $d \cdot 2^{d-1}$ edges, diameter d, and the broadcast time of d under the classical model is achieved as follows: The originator learns the message at time 0. Each vertex α gets informed at time $t \le d - 1$

and will call its neighbours $\alpha(t), \cdots, \alpha(d-1)$ at time $t+1, t+2, \cdots, d$, respectively. Figure 2.10 shows H_3.

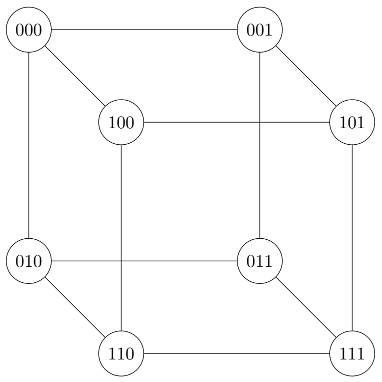


Figure 2.10: Hypercube with $d = 3$

2.5.11 Cube-Connected Cycle CCC_d

The Cube Connected Cycle CCC_d of dimension d is formed by replacing every vertex of a $d-$dimensional Hypercube (H_d) with a cycle of length d. In particular, the set of vertices of a CCC_d is as follows: $V_{CCC_d} = \{0,1\}^d \times \{0, 1, \cdots, d-1\}$, in which $\{0,1\}^d$ denotes the set of binary strings with length d. Thus, a vertex $v \in V_{CCC_d}$ is represented by a pair $v = \langle \alpha, i \rangle, \alpha \in \{0,1\}^d$, and $i \in \{0, 1, \cdots, d-1\}$, $\alpha = a_0 a_1 \cdots a_{d-1}$, where $a_i \in \{0,1\}$, $0 \leq i \leq d-1$. The edges of CCC_d are of three types: For each $i \in \{0, 1, \cdots, d-1\}$ and each $\alpha = a_0 a_1 \cdots a_{d-1} \in \{0,1\}^d$, the vertex $\langle \alpha, i \rangle$ is connected with a Forward port: $\langle \alpha, i+1 \pmod d \rangle$, a Backward port: $\langle \alpha, i-1 \pmod d \rangle$, and a Lateral port: $\langle \alpha(i), i \rangle$, where $\alpha(i) = a_0 a_1 \cdots a_{i-1} \bar{a}_i a_{i+1} \cdots a_{d-1}$. Note that $\bar{a}$ represents the binary complement of a. We call $\alpha(i)$ the ith dimensional neighbour of α. A CCC_d has $d \cdot 2^d$ vertices, $3d \cdot 2^{d-1}$ edges, diameter of $\lfloor \frac{5d}{2} \rfloor - 2$ for any $d > 3$, and a broadcast time of $\lceil \frac{5d}{2} \rceil - 1$ under the classical model. Figure 2.11 shows CCC_3.

2.5.12 Shuffle Exchange SE_d

The Shuffle-Exchange network SE_d of dimension d has the following set of vertices: $V_{SE_d} = \{0,1\}^d$, which denotes the set of binary strings with length d. The edges of SE_d are of two types: shuffle edges $(\alpha a, a\alpha)$, and exchange edges $(\alpha a, \alpha\bar{a})$, where $\alpha \in \{0,1\}^{k-1}, a, \bar{a} \in \{0,1\}, a \neq \bar{a}$.

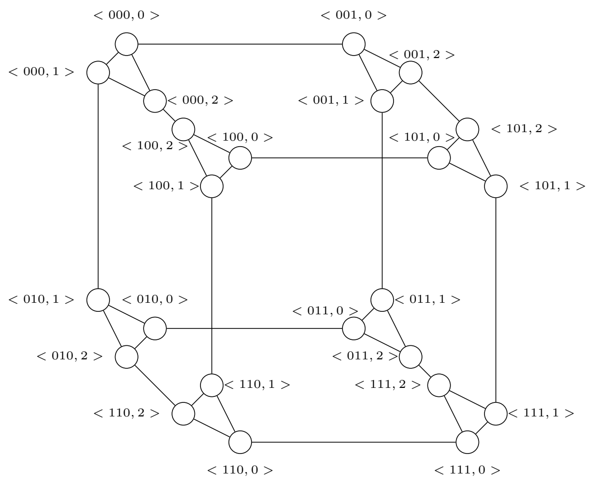


Figure 2.11: Cube-Connected Cycle with $d = 3$

The SE_d has 2^d vertices, a diameter of $2d-1$, and a maximum degree of 3. Also, $B_{cl}(SE_d) = 2d-1$ (Hromkovič et al., 1993). Figure 2.12 shows SE_3.

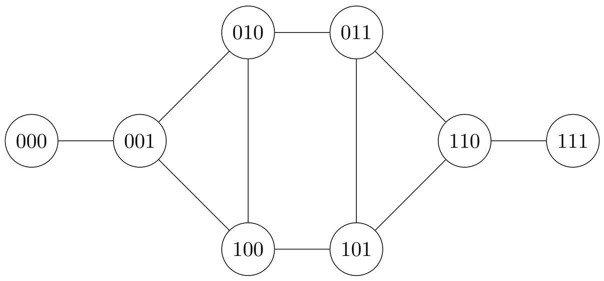


Figure 2.12: Shuffle Exchange network with $d = 3$

2.5.13 DeBruijn DB_d

The DeBruijn network DB_d of dimension d has the following set of vertices: $V_{DB_d} = \{0, 1\}^d$, which denotes the set of binary strings with length d. The edges of DB_d are of two types: shuffle edges $(\alpha a, a\alpha)$, and shuffle-exchange edges $(a\alpha, \alpha\bar{a})$, where $\alpha \in \{0, 1\}^{k-1}, a, \bar{a} \in \{0, 1\}, a \neq \bar{a}$. The BD_d has 2^d vertices, a diameter of m, and a maximum degree of 4. The value of the classical broadcast time of BD_d is not known. The tightest bounds known are as follows: $1.3171d \leq B_{cl}(BD_d) \leq \frac{3}{2}(d+1)$ (Bermond & Peyrat, 1988; Klasing, Monien, Peine, & Stöhr, 1994). Figure 2.13 shows BD_3.

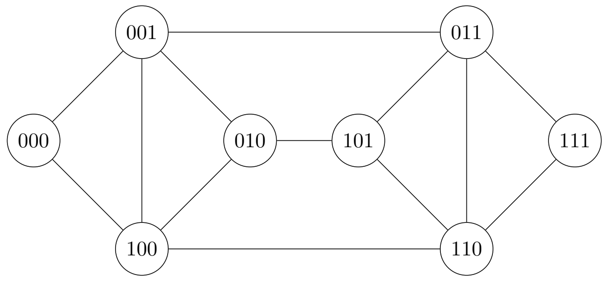


Figure 2.13: DeBruijn network with $d = 3$

2.5.14 Harary graph $H_{k,n}$

A Harary graph is defined by Frank Harary as follows (Harary, 1962):

(1) k **is even:** Let $k = 2r$. Then, $H_{2r,n}$ is constructed as follows: Given two positive integers n and $2r$ with $2r \leq n$, draw an $n-$gon and label its points with $0, 1, \cdots, n-1$. If $|i-j| \equiv m \mod n$, add an edge between i and j, where $1 \leq m \leq r$. Figure 2.14 shows $H_{4,12}$.

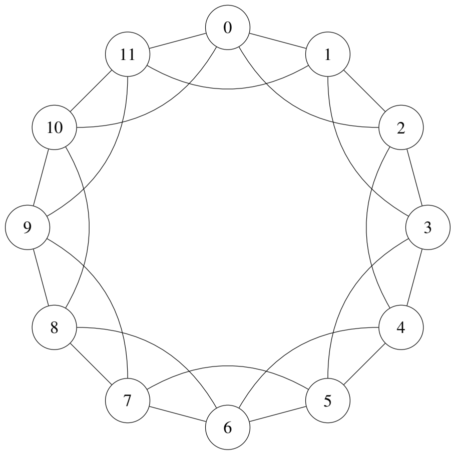


Figure 2.14: Harary graph with $k = 4, n = 12$

(2) k **is odd,** n **is even:** Let $k = 2r+1$. Then, $H_{2r+1,n}$ is constructed by drawing $H_{2r,n}$ and then adding edges by joining vertex i to vertex $i + \frac{n}{2}$ for $0 \leq i \leq \frac{n}{2} - 1$. Figure 2.15 shows $H_{5,12}$.

(3) k **and** n **are odd:** Let $k = 2r+1$. Then, $H_{2r+1,n}$ is constructed by drawing $H_{2r,n}$ and then adding edges by joining vertex 0 to vertices $\frac{n-1}{2}$ and $\frac{n+1}{2}$ and vertex i to vertex $i + \frac{n+1}{2}$ for $0 < i < \frac{n-1}{2}$. Figure 2.16 shows $H_{5,13}$.

In all cases, n is sufficiently greater than k. $H_{k,n}$ is a k-connected graph on n vertices with a degree

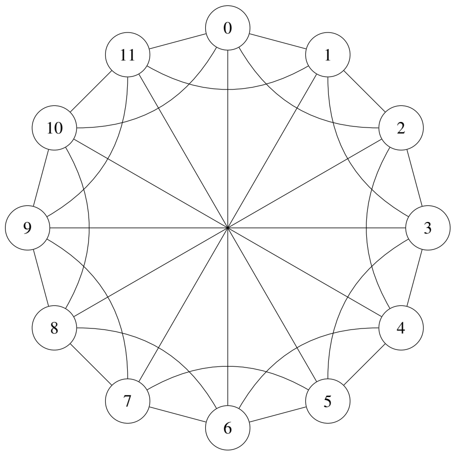


Figure 2.15: Harary graph with $k = 5, n = 12$

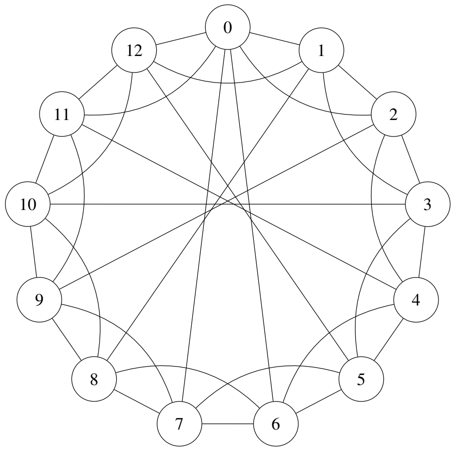


Figure 2.16: Harary graph with $k = 5, n = 13$

of at least k and $\lceil \frac{kn}{2} \rceil$ edges. The problem of broadcasting in this family of networks and graphs similar to Harary graphs are studied in (Bhabak et al., 2017, 2014).

Chapter 3

Optimal Broadcasting in Fully Connected Trees

3.1 Introduction

This chapter presents an optimal algorithm for Fully Connected Trees (FCT_n). A Complete graph K_n is a graph with n vertices and all possible edges. A tree T is a connected acyclic graph in which there exists exactly one path between any two vertices. An FCT_n is constructed using these two families:

Notation 3.1.1. *The vertices and the edges of a complete graph K_n are denoted by $V_{K_n} = \{1, 2, 3, \cdots, n\}$, and $E_{K_n} = \{(i, j) | i \neq j$ and $i, j \leq n\}$, respectively.*

Notation 3.1.2. *Consider n trees with their corresponding roots. We denote the trees by $T_i = (V_i, E_i)$ where $1 \leq i \leq n$, and the roots by: $\{1, 2, 3, \cdots, n\}$. The set of vertices and edges of all trees are denoted by V_T and E_T, respectively:*

$$
\begin{aligned}
V_T &= \bigcup_{i=1}^{n} V_i \\
E_T &= \bigcup_{i=1}^{n} E_i
\end{aligned}
\tag{3}
$$

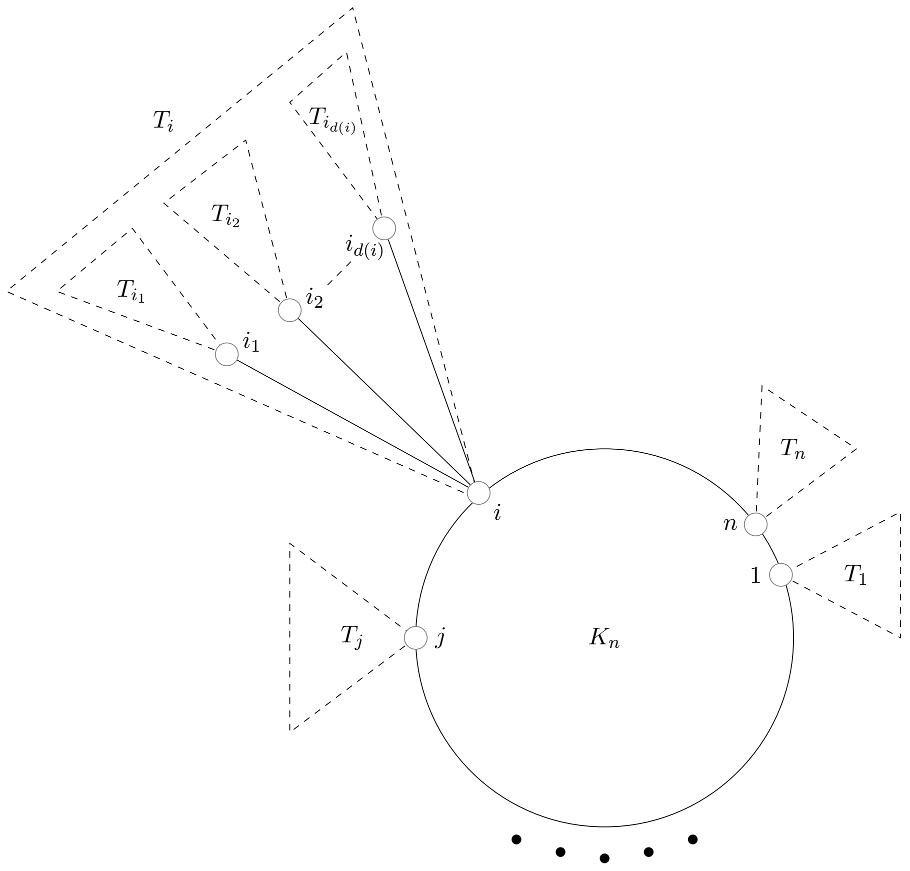


Figure 3.1: A Fully Connected Tree FCT_n

Definition 3.1.1. *Consider a complete graph K_n in a way that each vertex is the root of a tree, the resulting graph is called a Fully Connected Tree of size n or $FCT_n = (V, E)$ in which $V = V_T$ and $E = E_T \cup E_{K_n}$.*

Definition 3.1.2. *The members of the set $V \cap V_{K_n}$ are called root vertices, while the vertices in $V \backslash V_{K_n}$ are referred to as tree vertices.*

Notation 3.1.3. *Each tree T_i consists of $d(i)$ sub-trees rooted at $i_1, \cdots, i_{d(i)}$ where $d(i)$ is the number of children of vertex i in the tree T_i i.e. $d(i) = d_i - n - 1$, where d_i is the degree of vertex i. These sub-trees are called $T_{i_1}, \cdots, T_{i_{d(i)}}$ such that $B_{cl}(i_1, T_{i_1}) \geq \cdots \geq B_{cl}(i_{d(i)}, T_{i_{d(i)}})$.*

These definitions and notations are shown in Figure 4.1.

The result of this chapter is an extension of the conference paper (Harutyunyan & Maraachlian, 2009a), which briefly presented a $O(|V| \log |V|)$ algorithm for finding the broadcast time of a Fully

Connected Tree without giving proof of correctness. The main contribution of this chapter is as follows. We propose the algorithm's details that find the exact broadcast time of a vertex in an FCT_n. This enables us to give a non-trivial proof of the correctness of the algorithm, which was not given in (Harutyunyan & Maraachlian, 2009a). We also improve the complexity and obtain a $O(|V| \log \log n)$ time algorithm to calculate the broadcast time of an arbitrary originator in an FCT_n.

The remainder of this chapter is organized as follows: In the next Section, we present a broadcast algorithm for fully connected trees where the originator is a *root vertex*. We also prove the algorithm's correctness and analyze its complexity. In Section 3.3, we consider the broadcast problem where the originator is a *tree vertex*.

3.2 A Broadcast Algorithm for Root Vertices

This section considers the broadcast problem in an FCT_n when the originator is a *root vertex*. We first establish lower and upper bounds on the broadcast time:

Lemma 3.2.1. *The broadcast time of an arbitrary root vertex u in an FCT_n is bounded as follows ($1 \leq i \leq n$):*

$$\underbrace{\max\left\{\lceil \log n \rceil, \max\{B_{cl}(i, T_i)\}\right\}}_{lb} \leq B_{cl}(u, FCT_n) \leq \underbrace{\lceil \log n \rceil + \max\{B_{cl}(i, T_i)\}}_{ub} \tag{4}$$

Proof. For the upper bound (ub), a trivial algorithm for broadcasting in an FCT_n suggests completing the broadcasting within the complete graph in the first $\lceil \log n \rceil$ time units. Afterward, all root vertices may start the broadcasting within their trees simultaneously, which will not take more than $\max\{B_{cl}(i, T_i)\}$ time units. Hence, $B_{cl}(u, FCT_n) \leq \lceil \log n \rceil + \max\{B_{cl}(i, T_i)\}$.

To prove the lower bound (lb), note that any algorithm for this problem must broadcast in the complete graph and all the trees. Regardless of the algorithm's procedure, it has to spend at least $\lceil \log n \rceil$ time units for the complete graph while informing all trees is impossible in less than $\max\{B_{cl}(i, T_i)\}$ time units. Thus, Any broadcast algorithm has to respect those bounds, resulting in a lower bound of $B_{cl}(u, FCT_n) \geq \max\left\{\lceil \log n \rceil, \max\{B_{cl}(i, T_i)\}\right\}$. □

3.2.1 Broadcast Algorithm

We reduce the problem of finding the broadcast time of a root vertex u in an FCT_n to a problem in which the aim is to study the feasibility of finishing the broadcasting in an FCT_n in τ time units starting from a particular root vertex, such as u. Based on **Lemma 3.2.1**, this problem must be solved only when $lb \leq \tau \leq ub$. Otherwise, if $\tau < lb$, it is impossible to find a broadcast scheme, and when $\tau > ub$, a trivial algorithm works. Having proposed an algorithm for the aforementioned problem, we solve the main problem with another algorithm that utilizes a method similar to the binary search for the interval of Equation (4). These algorithms are called B_{Search} and BR_τ, which will be discussed in detail.

BR_τ takes a Fully Connected Tree FCT_n, an originator u, and a candidate broadcast time τ. It is supposed to return TRUE if $br(u, FCT_n) \leq \tau$, and FALSE otherwise. In other words, only if BR_τ finds a broadcast scheme that completes the message distribution in the input FCT_n within τ time units will it return TRUE. If the output of BR_τ for a particular τ is FALSE, it means that $B_{cl}(u, FCT_n) > \tau$. So, we need to invoke the algorithm for a larger value of τ in order to succeed. However, in the case of TRUE output, one can conclude that $B_{cl}(u, FCT_n) \leq \tau$. In order to argue the exact broadcast time, not only should BR_τ return TRUE for τ, but it must also output FALSE for $\tau - 1$.

Instead of going over all the possible values of Equation (4) as the candidate broadcast time τ, B_{Search} is deployed. It follows a similar procedure to binary search to reduce the size of the range iteratively. A pseudo-code for B_{Search} is provided in the Algorithm 1. We start from the mid-point of lb and ub. If a broadcast scheme is founded, the search will continue on the lower half with the updated upper bound; otherwise, we will search the other half while updating the lower bound. The initial call for this algorithm is $B_{Search}(FCT_n, u, lb, ub)$ where lb and ub come from Equation (4). BR_τ that is invoked throughout Algorithm 1 is described hereafter.

The first step of BR_τ is to assign weights to every *root vertex* of the fully connected tree, $w(i, t)$, and calculate m_{i_j}'s ($1 \leq i \leq n, 1 \leq j \leq d(i)$) (Line 1 in Algorithm 2). These weights are the labels that the algorithm in (Slater et al., 1981) assigns to the vertices of a tree. If vertex i does not have any uninformed children in T_i, then its weight is zero because i can do nothing to speed up the broadcast

Algorithm 1 The modified Binary Search algorithm $B_{Search}(FCT_n, u, lb, ub)$

```
    Input: $FCT_n = (V, E)$, originator $u$, lower bound $lb$, and upper bound $ub$.
    Output: Broadcast time $\tau$ such that $\tau = B_{cl}(u, FCT_n)$
 1: $t = lb + \lfloor \frac{ub-lb}{2} \rfloor$;
 2: if $lb == ub$ then
 3:     if $BR_\tau(FCT_n, u, lb)$ then
 4:         return $lb$
 5:     end if
 6: end if
 7: if $lb + 1 == ub$ then
 8:     if $BR_\tau(FCT_n, u, lb)$ then
 9:         return $lb$
10:     end if
11:     if $!BR_\tau(FCT_n, u, lb)$ and $BR_\tau(FCT_n, u, ub)$ then
12:         return $ub$
13:     end if
14: end if
15: if $BR_\tau(FCT_n, u, t)$ then
16:     return $B_{Search}(FCT_n, u, lb, t)$
17: else
18:     return $B_{Search}(FCT_n, u, t, ub)$
19: end if
```

process in T_i. Whereas, if i has one or more uninformed children, its weight will equal the time needed to complete the broadcasting in its sub-trees. The weight of each root vertex i is initialized to the broadcast time, so for $t = 0$; $w(i, t) = B_{cl}(i, T_i)$. Additionally, m_{i_j} is the time needed to finish the broadcasting in T_{i_j} originating at *tree vertex* i_j, $m_{i_j} = B_{cl}(i_j, T_{i_j})$. Note that using the algorithm provided in (Slater et al., 1981) for a *root vertex* i, its children $i_1, \cdots, i_{d(i)}$ will be arranged in a way that $m_{i_1} \geq \cdots \geq m_{i_{d(i)}}$. Thus, for $t \geq 1$, $w(i, t)$ could be calculated by its corresponding m_{i_j} labels utilizing the logic provided in (Slater et al., 1981): $w(i, t) = \max_{1 \leq j \leq d(i)}(j + m_{i_j})$. Also, the set of informed vertices (V_I) and uninformed vertices (V_U) are initialized at the beginning of the algorithm. Lastly, assuming u is the originator, l_u is set to NULL (Line 2 in Algorithm 2).

Another label that is defined for a *root vertex* i is l_i, which is the remaining time for an uninformed *root vertex* i until it must get informed; otherwise, it will not be able to inform its *tree vertices* in τ time units. This label is assigned to: $l_i = \tau - t - w(i, t) - 1$. The logic behind this definition is simple: the total remaining time is $\tau - t$. However, it will take $w(i, t)$ time units to complete the broadcasting in T_i. Besides, since in a complete graph, the distance between any pair of vertices is 1, it will take exactly a single unit of time to send the message to i, regardless of prior knowledge about the set of informed vertices. We set $\forall i \in V_I : l_i$=NULL. But the value of l_i will be updated for all uninformed *root vertices* at each time unit based on the definition. Clearly, an uninformed *root vertex* with a smaller value of l_i has a higher priority and must be informed before other uninformed *root vertices*. It should be mentioned that for any $i \in V_U$ if $l_i < 0$ at any time unit, i cannot complete broadcasting in τ time units within its tree T_i.

At each time unit t, $0 \leq t \leq \tau$, all vertices of the FCT_n are considered for performing the most appropriate action. If at some point, a *root vertex* notices that τ is not enough for completing the broadcasting in the graph, the algorithm terminates and outputs FALSE. If that does not happen, we will continue the message-passing process, and each informed vertex will communicate to one of its uninformed neighbors. This process will continue as long as all vertices of the FCT_n are informed or a vertex interrupts the execution with a FALSE output.

During each time unit t, if the considered vertex v is an uninformed root vertex, we only update the value of l_v (Line 6 in Algorithm 2). On the other hand, if v is already informed, the algorithm has to decide on the action that v must take. If v is a *tree vertex*, the optimal decision is to follow the well-known broadcast algorithm in trees (Slater et al., 1981) (Lines 29-31 of Algorithm 2). However, when v is a *root vertex*, there are two options to choose from: Either to send the message to its tree T_v or to contribute to the complete graph K_n and send the message to an uninformed root vertex. In the following, we discuss how to choose between these two options:

(1) If $w(v,t) = 0$, it means that vertex v does not have uninformed children in T_v. In this case, v should inform an uninformed *root vertex* with the lowest value of l_i, if such vertex exists,

(2) If $w(v,t) > 0$, v should choose between the two mentioned options. This is done by comparing the time needed to inform the uninformed tree attached to it, $w(v,t)$, with the remaining time $\tau - t$:

- If $w(v,t) < \tau - t$, it means that v still has more than enough time to complete the broadcasting within its tree. Therefore, it informs another uninformed *root vertex* with the smallest value of l_i if such vertex exists (Lines 13-14 of Algorithm 2). If not, v

Algorithm 2 The broadcast algorithm $BR_\tau(FCT_n, u, \tau)$

```
**Input:** $FCT_n = (V, E)$, originator $u$, candidate broadcast time $\tau$.
**Output:** FALSE if $\tau$ cannot be the broadcast time, TRUE if broadcasting can be accomplished in at most $\tau$ time units.
1: **Initialize:** the labels $w(i, t)$ and $m_{i_j}$ for all *root vertices*;
2: **Initialize:** $V_I = \{u\}$, $V_U = V \backslash V_I$, $l_u$ =NULL;
3: **for each** $t$ such that $0 \leq t \leq \tau - 1$ **do**
4:     **for each** $v \in V_U$ **do**
5:         **if** $v$ is a *root vertex* **then**
6:             **update** $l_v$ as follows: $l_v = \tau - t - w(v, t) - 1$;
7:         **end if**
8:     **end for**
9:     **for each** $v \in V_I$ **do**
10:        **if** $v$ is a *root vertex* **then**
11:            **if** $w(v, t) < \tau - t$ **then**
12:                **if** there exists at least one uninformed root vertex **then**
13:                    $v$ informs vertex $j$ at time $t$ such that $j$ has the smallest value of $l_\alpha$ in $V_U$;
14:                    $l_j$=NULL, $V_I = V_I \cup \{j\}$, $V_U = V_U \backslash \{j\}$;
15:                **else**
16:                    $v$ stays idle;
17:                **end if**
18:            **else**
19:                **if** $w(v, t) = \tau - t$ **then**
20:                    $v$ informs one of its children which has the highest value of $m_{v_j}$ in the
21:                    tree rooted at $T_v$, $1 \leq j \leq d(v)$;
22:                    $m_{v_j}$=NULL, $V_I = V_I \cup \{v_j\}$, $V_U = V_U \backslash \{v_j\}$;
23:                    **update** $w(v, t) = \max_{1 \leq k \leq d(v)}\{k + m_{v_k}\}$;
24:                **else**
25:                    **return** FALSE;
26:                **end if**
27:            **end if**
28:        **else**
29:            $v$ informs a *tree vertex* $v_T$ in the uninformed sub-tree rooted at $v$ based on the
30:            well-known broadcasting algorithm in trees;
31:            $V_I = V_I \cup \{v_T\}, V_U = V_U \backslash \{v_T\}$;
32:        **end if**
33:    **end for**
34: **end for**
35: **return** TRUE;
```

remains idle until time unit t' for which $\tau - t' = w(v, t')$ (Line 16 of Algorithm 2).

- If $w(v, t) = \tau - t$ then v has to inform one of its children within its tree T_v according to the broadcast algorithm in trees. This vertex must have the highest value of m_{v_j} in all sub-trees rooted at v, $1 \leq j \leq d(v)$ (Lines 20-23 of Algorithm 2).

- Finally, if $w(v,t) > \tau - t$, it means that τ cannot be the broadcast time since v is not able to finish the broadcasting even within its tree T_v. So, the algorithm returns FALSE (Line 25 of Algorithm 2).

The details of algorithm BR_τ are presented in Algorithm 2. It is worth mentioning that when several *root vertices* have the same value of l_i, the algorithm randomly chooses a vertex to proceed. It means that either there are several optimal broadcast schemes or broadcasting cannot be finished with the given τ. Lastly, we describe the operations carried out by BR_τ with a small example. Consider a fully connected tree with 5 trees, T_i where $1 \leq i \leq 5$. The originator is vertex 1, the root of the tree T_1, and the candidate broadcast time is given as $\tau = 6$.

Figure 6.1 represents the graph and the whole series of calls made throughout the time $t = 1$ till $t = 6$. In the following, it is described how the algorithm initiates each call. In the beginning, since $w(1,0) = \tau - t = 6$, vertex 1 cannot do anything but inform one of its children based on the broadcasting algorithm in trees. Afterward, the labels are updated, as can be seen in Table 7.1. The changes of the labels compared to that of $t = 0$ are shown with crossed lines. Firstly, at this time unit, m_{1_1} will be set to NULL since the root of T_{1_1} is already informed. Then, $w(1,t)$ will be updated. Moreover, l_i should be recalculated for all the *root vertices* in V_U.

At time $t = 2$, vertex 1 has to decide on its next call. Since $w(1,t) < \tau - t$, this vertex does not need to continue broadcasting within its tree for now. However, it must choose one of the *root vertices* with the smallest value of l_i. Based on Table 7.1, it selects vertex 2 for continuing the process. After this call, the labels must be updated again. The only changes are as follows: l_2 will be set to NULL since vertex 2 is already informed. Also, l_3, l_4, and l_5 decrease by 1 and are set to 1,3 and 0, respectively. The rest of the table remains the same.

At time $t = 3$, vertices 1 and 2 should decide on their next call. Since $w(1,t) < \tau - t$, vertex 1 continues to broadcast within the complete graph. It chooses 5 since it has the minimum value of l_i. But $w(2,t) = \tau - t$, so vertex 2 will broadcast within its tree. This process will continue until all the vertices of the mentioned graph are informed within $\tau = 6$ time units (See Figure 6.1). Hence,

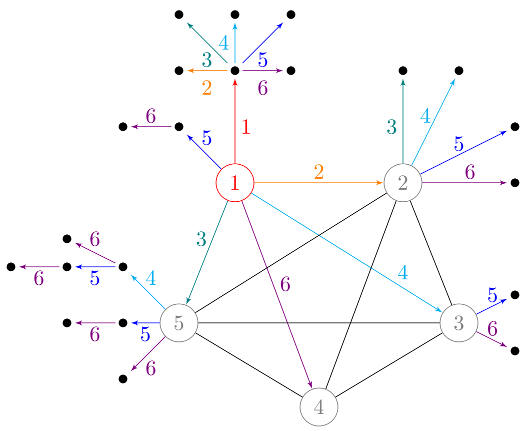


Figure 3.2: A fully connected tree FCT_5 with 5 trees T_i rooted at i, $1 \leq i \leq 5$, and the originator vertex 1. The series of calls are shown with colorful arrows.

Labels after $t = 1$						
root vertex i	$w(i,t)$	m_{i_1}	m_{i_2}	m_{i_3}	m_{i_4}	l_i
1	$\cancel{6}$ 2	$\cancel{5}$ -	1	-	-	-
2	4	0	0	0	0	$\cancel{1}$ 0
3	2	0	0	-	-	$\cancel{3}$ 2
4	0	-	-	-	-	$\cancel{5}$ 4
5	3	2	1	0	-	$\cancel{2}$ 1

Table 3.1: The labels for the example graph after $t = 1$

it is concluded that $B_{cl}(1, FCT_5) \leq 6$. To argue whether $B_{cl}(1, FCT_5) = 6$ or not, Algorithm 1 has to run on the lower and upper bound of the broadcast time of the example graph. Since time $\tau = 6$ matches the lb for this graph, one may conclude that $B_{cl}(1, FCT_5) = 6$.

It should be noted that the way the algorithm is presented in (Harutyunyan & Maraachlian, 2009a) fails to inform all the vertices of this example in 6 time units starting from 1. This is because a root vertex i will not stop broadcasting within T_i unless all of its children in T_i are informed. It means that for the graph illustrated in Figure 6.1, vertex 1 will initiate two calls to its children in the first and the second time units since $w(1, 0) = \tau - t$. At time $t = 3$, however, since $w(2, t) > \tau - t$, the algorithm concludes that the time $\tau = 6$ is not enough for broadcasting in FCT_5 and returns FALSE.

The same pattern will happen for FCT_n's, which have a huge sub-tree with a large broadcast time rooted at i_1, and many small sub-trees rooted at i_j's ($1 < j \leq d(i)$). The algorithm in

(Harutyunyan & Maraachlian, 2009a) will complete the broadcasting in T_i, while it is not necessary to inform all i_j's at the first $d(i)$ time units, since $w(i,t)$ could decrease significantly as soon as i_1 is informed (as an example, note that in Table 7.1, $w(1,t)$ decreases from 6 to 2 with only one call within T_1). The optimal decision for vertex i is to inform i_1 at time $t=1$, and afterward, it could forget about $i_2, \cdots, i_{d(i)}$ for some time units and come back to its sub-trees only when necessary.

3.2.2 Proof of Correctness

This section proves the correctness of the =broadcast algorithm presented earlier. We do so by establishing two statements: first, if $BR_\tau(FCT_n, u, \tau)$ outputs TRUE, there is a broadcast scheme in which all the vertices of the FCT_n are informed within τ time units, and the algorithm finds this scheme (Theorem 3.2.1). Secondly, if the algorithm outputs FALSE, it is impossible to inform all vertices of FCT_n within τ time units using any other broadcast scheme originating from vertex u (Theorem 3.2.2). From Theorems 3.2.1 and 3.2.2, one can conclude the correctness of the proposed algorithm.

Theorem 3.2.1. *If Algorithm 2, $BR_\tau(FCT_n, u, \tau)$, outputs TRUE, then, $B_{cl}(u, FCT_n) \leq \tau$.*

Proof. It is enough to show that when Algorithm 2 outputs TRUE, then $\forall i : 1 \leq i \leq n$, vertex i receives the message at time t'_i where $w(i, t'_i) \leq \tau - t'_i$. It immediately follows that all the *root vertices* have received the message when it is enough for them to inform all of their *tree vertices* within the remaining time $\tau - t'_i$.

In Algorithm 2, the output is TRUE when the FALSE condition has not happened during $0 \leq t \leq \tau - 1$. In other words, when a *root vertex* i is not in a situation where $w(i,t) > \tau - t$ (following the pseudo-code of Algorithm 2 at line 25). Thus, it is easy to see that when BR_τ outputs TRUE, it gives a broadcast scheme in which all the *root vertices* are provided with enough time to complete the broadcasting within their corresponding sub-trees. When all the *root vertices* are informed, and they have enough remaining time to inform all their children, it is concluded that the broadcasting is completed in the FCT_n. Therefore, the broadcast scheme generated by Algorithm 2 is a valid broadcast scheme for the originator u in the input graph FCT_n. □

Now, it will be proved that if the proposed algorithm outputs FALSE, no other algorithm can complete the broadcasting within τ time units.

Assume an infinitely large complete graph is given for studying the broadcasting from one of its vertices. Denote the set of informed vertices at time t by N_t. The goal is to calculate the number of informed vertices at time $t + \Delta$ with two conditions:

(1) There are α different vertices (out of N_t) that will stay idle for one time unit. In other words, these vertices will not inform a new vertex in the complete graph during one time unit.

(2) The idle time unit can be anytime in the time interval $[t, t+\Delta]$ where Δ is any positive integer.

Lemma 3.2.2. *The number of informed vertices at time $t + \Delta$ will be maximized if all of the α vertices choose to remain idle at time $t + \Delta$.*

Proof. First, assume one vertex is idle at time $t + r$, where $r \in [0, \Delta]$ during broadcasting. Then, the total number of informed vertices at time $t + \Delta$ and before is calculated as shown in Table 3.2. Note that the number of informed vertices will double during time units $t, \cdots, t + \Delta$, except for time $t + r$.

time	number of informed vertices
t	$\|N_t\|$
$t+1$	$2\|N_t\|$
$t+2$	$2^2\|N_t\|$
$\cdots$	$\cdots$
$t+r-1$	$2^{r-1}\|N_t\|$
$t+r$	$2^{r-1}\|N_t\| + 2^{r-1}\|N_t\| - 1 = 2^r\|N_t\| - 1$
$t+r+1$	$2(2^r\|N_t\| - 1) = 2^{r+1}\|N_t\| - 2$
$\cdots$	$\cdots$
$t+r+i$	$2^i(2^r\|N_t\| - 1) = 2^{r+i}\|N_t\| - 2^i$
$\cdots$	$\cdots$
$t+\Delta-1$	$2^{\Delta-1-r}(2^r\|N_t\| - 1) = 2^{\Delta-1}\|N_t\| - 2^{\Delta-1-r}$
$t+\Delta$	$2^{\Delta}\|N_t\| - 2^{\Delta-r}$

Table 3.2: The changes in the number of informed vertices following Lemma 3.2.2.

Now the general case is considered, when the α vertices decide to stay idle at the time units $t + t'_1, \cdots, t + t'_\alpha$, where $0 \leq t'_j \leq \Delta$ for $j = 1, \cdots, \alpha$. Then the number of informed vertices at time $t + \Delta$ is $|N_{t+\Delta}| = 2^{\Delta}|N_t| - 2^{\Delta-t'_1} - \cdots - 2^{\Delta-t'_\alpha}$. Observe that $|N_{t+\Delta}|$ will be maximized when $t'_j = \Delta$ for all $1 \leq j \leq \alpha$, which is equal to $2^{\Delta}|N_t| - \alpha$. Therefore, the number of informed vertices will be maximized if all of those α vertices choose to remain idle at the last time unit. □

It is worth pointing out the relevance of **Lemma 3.2.2** to the problem of broadcasting in fully connected trees. The idle time unit of a vertex in the above lemma could correspond to a *root* vertex either spending one time unit informing a *tree* vertex or remaining idle. Consequently, **Lemma 3.2.2** will be used for showing that it is necessary for a *root* vertex to avoid broadcasting within its sub-tree or even staying idle any time sooner than $t = \tau - w(i, t)$. In other words, as long as a *root* vertex i is able to delay the start of broadcasting within its subtree T_i, it must do it and continue the broadcasting in the complete graph.

Lemma 3.2.3. *Consider an arbitrary FCT_n and fix the originator u. Assume that $B_{cl}(u, FCT_n) = \tau$. If under broadcast scheme S_τ, a root vertex i informs a tree vertex i_j at time t_1, then under any other scheme S in the same graph achieving broadcast time τ, it is necessary to inform vertex i_j by the time t_1.*

Proof. Denote the broadcast time of originator u in FCT_n under broadcast scheme S by $B_{cl_S}(u, FCT_n)$. Based on the proposed algorithm, an informed *root vertex* i has 2 options in each time unit t_1: if $w(i, t_1) < \tau - t_1$ then it informs another *root vertex*, while if $w(i, t_1) = \tau - t_1$, the *root vertex* i informs a *tree vertex* with the maximum value of m_{i_j}. This *tree vertex* is denoted by i_j. Let i_j be informed at time t' under broadcast scheme S. The goal is to show that when $w(i, t_1) = \tau - t_1$, vertex i_j must have been informed by the time $t' \leq t_1$ in any other scheme S with $B_{cl_S}(u, FCT_n) \leq \tau$.

By contradiction, suppose that $t' > t_1$, meaning that vertex i initiates a call to inform vertex i_j under broadcast scheme S at time $t' > t_1$. Firstly, note that since the proposed algorithm chooses the *tree vertex* with the maximum value of m_{i_j} at each time unit, it is concluded that the sub-tree T_{i_j} has the biggest weight among all sub-trees with uninformed *root vertices* in T_i. Hence, for decreasing the value of $w(i, t)$ for any t, one needs to initiate a call from i to i_j. Furthermore, at time t_1, $w(i, t_1) = \tau - t_1$. Clearly, for any $t > t_1$, $\tau - t_1$, or the remaining time, decreases, while $w(i, t_1)$, or the time needed to finish broadcasting within T_i remains the same unless a call is made from i to i_j. This is because vertex i_j was not informed by the time t_1. Thus, broadcasting cannot finish in τ time units under the broadcast scheme S , or $B_{cl_S}(u, FCT_n) > \tau$. This contradicts the assumption.

Therefore, it is concluded that when at time t_1 a call is initiated from a *root vertex* to a *tree vertex* in S_τ, any broadcast scheme S must have informed that particular vertex not later than t_1. $\square$

Definition 3.2.1. *The set of informed root vertices by the time t under the broadcast scheme S is denoted by $V_t(S)$.*

Lemma 3.2.4. *Assume $B_{cl}(u, FCT_n) = \tau$. Let S_{opt} be an optimum broadcast scheme different than S_τ. Then, at any time t, $|V_t(S_\tau)| \geq |V_t(S_{opt})|$.*

Proof. This lemma will be proved by induction on t. For the base case, one can easily argue that $|V_0(S_\tau)| \geq |V_0(S_{opt})|$ since at time $t = 0$ only the originator is informed in both schemes. Now, assume $|V_i(S_\tau)| \geq |V_i(S_{opt})|$, it should be proved that $|V_{i+1}(S_\tau)| \geq |V_{i+1}(S_{opt})|$. The set $V_i(S_\tau)$ could be broken down into three disjoint sets:

(1) $V_i^{K_n}(S_\tau)$: the set of informed *root vertices* that have to inform another *root vertex* at time i following scheme S_τ.

(2) $V_i^{T}(S_\tau)$: the set of informed *root vertices* that have to inform a *tree vertex* at time i following scheme S_τ. The set of *tree vertices* that are informed by a call from the members of $V_i^{T}(S_\tau)$ will be denoted by $V_i^{T'}(S_\tau)$. Observe that all the vertices in $V_i^{T'}(S_\tau)$ are *tree vertices*, hence, there is exactly one vertex in $V_i^{T}(S_\tau)$ that initiated a call to inform each particular vertex of $V_i^{T'}(S_\tau)$. Therefore, $|V_i^{T}(S_\tau)| = |V_i^{T'}(S_\tau)|$.

(3) $V_i^{idle}(S_\tau)$: the set of informed *root vertices* that remained idle at time i following scheme S_τ.

The definitions are shown in Figure (3.3). In this figure, vertices j, m, and p form $V_i^{T}(S)$ as they broadcast within their corresponding trees, and the vertices that are informed by a call from the members of $V_i^{T}(S)$ are those which belong to $V_i^{T'}(S)$. Vertices 1 and n belong to $V_i^{K_n}(S)$ since they contribute to the complete graph. Vertices k and f remain idle, so they form $V_i^{idle}(S)$.

Using the same notation, $V_i(S_{opt})$ could be broken down into three disjoint sets: $V_i^{K_n}(S_{opt})$, $V_i^{T}(S_{opt})$, and $V_i^{idle}(S_{opt})$. The cardinality of the aforementioned sets is as follows:

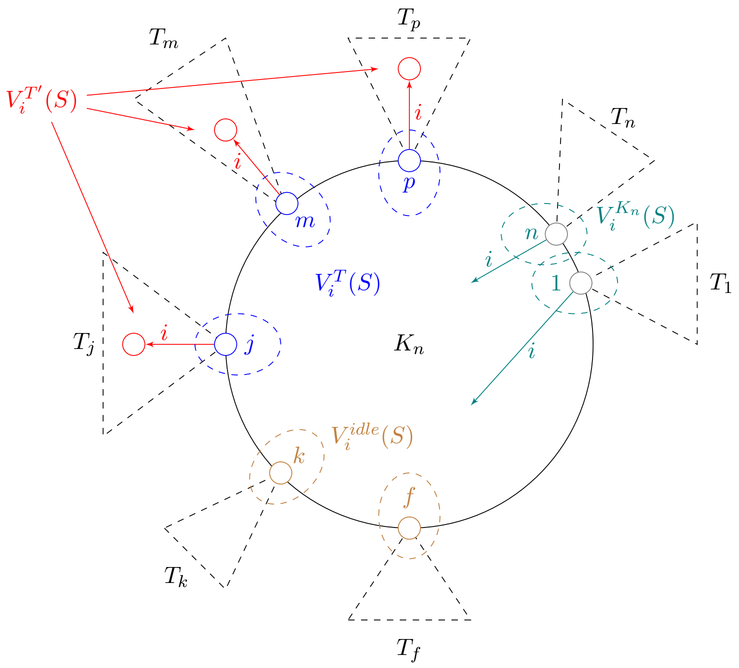


Figure 3.3: A Fully Connected Tree FCT_n, and a series of calls at time i under an arbitrary broadcast scheme S.

$$
\begin{aligned}
|V_i(S_\tau)| &= |V_i^{K_n}(S_\tau)| + |V_i^T(S_\tau)| + |V_i^{idle}(S_\tau)|, \\
|V_i(S_{opt})| &= |V_i^{K_n}(S_{opt})| + |V_i^T(S_{opt})| + |V_i^{idle}(S_{opt})|.
\end{aligned} \tag{5}
$$

At time $i+1$ under S_τ, the vertices of $V_i^{K_n}(S_\tau)$ will double, and the rest of the informed root vertices will remain informed. Thus:

$$
\begin{aligned}
&|V_{i+1}(S_\tau)| = |V_i^T(S_\tau)| + |V_i^{idle}(S_\tau)| + 2 \times |V_i^{K_n}(S_\tau)| \xrightarrow{\text{based on (5)}} \\
&|V_{i+1}(S_\tau)| = |V_i^T(S_\tau)| + |V_i^{idle}(S_\tau)| + 2 \times \Big(|V_i(S_\tau)| - |V_i^T(S_\tau)| - |V_i^{idle}(S_\tau)|\Big) \rightarrow \\
&|V_{i+1}(S_\tau)| = 2 \times |V_i(S_\tau)| - |V_i^T(S_\tau)| - |V_i^{idle}(S_\tau)|
\end{aligned} \tag{6}
$$

Similarly, the number of informed *root vertices* at time $i+1$ under S_{opt} is calculated as follows:

$$
|V_{i+1}(S_{opt})| = 2 \times |V_i(S_{opt})| - |V_i^T(S_{opt})| - |V_i^{idle}(S_{opt})| \tag{7}
$$

Now we calculate the difference between the number of informed *root vertices* in both schemes at time $i+1$:

$$|V_{i+1}(S_\tau)| - |V_{i+1}(S_{opt})| =$$
$$2 \times \Big(\underbrace{|V_i(S_\tau)| - |V_i(S_{opt})|}_{(1)}\Big) + \Big(\underbrace{|V_i^T(S_{opt})| - |V_i^T(S_\tau)|}_{(2)}\Big) + \Big(\underbrace{|V_i^{idle}(S_{opt})| - |V_i^{idle}(S_\tau)|}_{(3)}\Big) \tag{8}$$

In Equation (8), term (1) is non-negative based on the inductive hypothesis. We must show that terms (2) and (3) are non-negative. By **Lemma 3.2.3**, all the vertices of $V_i^{T'}(S_\tau)$ must be informed by any other broadcast scheme (including S_{opt}) by the time t' where $t' \leq i$. Two cases may arise:

(1) **All of the vertices in $V_i^T(S_\tau)$ broadcast within their trees under S_{opt} at time $t' = i$:** it immediately follows that $V_i^T(S_\tau) \subseteq V_i^T(S_{opt})$, and, $|V_i^T(S_{opt})| \geq |V_i^T(S_\tau)|$.

(2) **Some (or all) of the vertices in $V_i^T(S_\tau)$ have completed the broadcasting within their trees in S_{opt} at time $t' < i$:** in our broadcast algorithm, we force all root vertices to broadcast within their trees at the latest possible time, or when $\tau - t = w(i,t)$. Based on the aforementioned argument, any optimal broadcast scheme, including S_{opt}, has to follow the same pattern; otherwise, it will not be an optimized scheme. Hence, none of the vertices in $V_i^T(S_\tau)$ are allowed to complete the broadcasting within their trees in a time $t' < i$. Therefore, by **Lemma 3.2.2**, S_{opt} cannot be an optimal scheme which contradicts the hypothesis of this lemma.

From these cases it is concluded that $|V_i^T(S_{opt})| \geq |V_i^T(S_\tau)|$, so the second term in Equation (8) is also non-negative. Lastly, observe that using a similar argument, $|V_i^{idle}(S_{opt})| \geq |V_i^{idle}(S_\tau)|$ due to **Lemma 3.2.2**. Therefore, the third term in Equation (8) is non-negative as well. Consequently, $|V_{i+1}(S_\tau)| - |V_{i+1}(S_{opt})| \geq 0$ or $|V_{i+1}(S_\tau)| \geq |V_{i+1}(S_{opt})|$. □

Theorem 3.2.2. *Given a graph FCT_n, an originator u, and a time τ, if $BR_\tau(FCT_n, u, \tau)$ returns FALSE then $B_{cl}(u, FCT_n) > \tau$.*

Proof. Assume by contradiction that there exists a scheme S such that all the vertices of FCT_n are informed within τ time units. Since $BR_\tau(FCT_n, u, \tau)$ returned FALSE, then following our algorithm BR_{τ}, there was a *root vertex* j which got informed at time t and $w(j,t) > t-\tau$. Since our algorithm informs *root vertices* with the highest weights first, we are guaranteed that at any time, including time $t-1$, the set of informed *root vertices* of the broadcast scheme S_{τ}, denoted by $V_{t-1}(S_{\tau})$, all have weights greater than or equal to $w(j,t)$. Since S is a broadcast scheme with broadcast time τ, all the vertices in $V_{t-1}(S_{\tau})$ should be informed at time $t-1$ and should be part of the set $V_{t-1}(S)$ because all of these vertices have weights greater than the remaining time. Therefore we conclude that $V_{t-1}(S_{\tau}) \subseteq V_{t-1}(S)$.

On the other hand, S, being a broadcast scheme, should have informed vertex j at time $t-1$ or earlier, since it has a weight greater than $\tau - t$, or $w(j,t-1) > \tau - t$. Therefore we can conclude that $V_{t-1}(S_{\tau}) \subset V_{t-1}(S)$ and $|V_{t-1}(S_{\tau})| < |V_{t-1}(S)|$ which contradicts **Lemma 3.2.4**. Therefore, it is concluded that such a scheme S cannot exist if $BR_{\tau}(FCT_n, u, \tau)$ returns FALSE. $\square$

3.2.3 Complexity Analysis

This section will study the complexity of the algorithms proposed in this chapter. Firstly, note that $B_{Search}(FCT_n, u, lb, ub)$ does a binary search for the broadcast time in the range of Equation (4) in **Lemma 3.2.1**. The complexity of a binary search algorithm is $O(\log N)$ where N is the number of values in the range that is being searched in. In our algorithm $N = ub - lb$. Secondly, every time verifying if a certain value in the range is less than, greater than, or equal to the desired value, the $BR_{\tau}(FCT_n, u, \tau)$ has to be executed. Hence, the complexity of Algorithm 1 is:

$$O(\text{Algorithm 2}) \times O(\log(ub - lb)) \tag{9}$$

We need to calculate each part now. First observe that from **Lemma 3.2.1**:

$$\begin{aligned} ub - lb &= \lceil \log n \rceil + \max\{B_{cl}(i,T_i)\} - \max\left\{\lceil \log n \rceil, \max\{B_{cl}(i,T_i)\}\right\} \rightarrow \\ ub - lb &= \min\left\{\lceil \log n \rceil, \max\{B_{cl}(i,T_i)\}\right\} \leq \lceil \log n \rceil \end{aligned} \tag{10}$$

Therefore, $ub - lb \in O(\log n)$, where n is the number of vertices in the complete graph K_n.

Now, assume that $|V|$ is the number of vertices of the graph FCT_n. $BR_{\tau}(FCT_n, u, \tau)$ can calculate its decision in a linear time because every *root vertex* has to calculate its weight and compare it with the remaining time. Once the weights are calculated at the beginning of the algorithm, updating the weights at every new time unit can be done in a constant time. Also, initializing the weights at the beginning of the algorithm is a linear operation in terms of the number of vertices in the graph because the tree broadcast algorithm runs in linear (Slater et al., 1981). Also, note that every *root vertex* has to calculate its weight and compare it with the remaining time. The number of times the weight of a *root vertex* changes is equal to the number of children a *root vertex* has in the tree attached to it. Therefore, the number of comparisons needed before one of the *root vertices* makes a decision is, at most, the total number of vertices that the *root vertices* have. This number is a linear function of the total number of vertices. Therefore, it is concluded that Algorithm 2 has a complexity of $O(|V|)$.

This implies that $B_{Search}(FCT_n, u, lb, ub)$ has a complexity of $O(|V| \log \log n)$.

3.3 A Broadcast Algorithm for Tree Vertices

This section aims to provide an algorithm for finding the broadcast time of FCT_n when the originator is a *tree vertex*.

3.3.1 Broadcast Algorithm

Assume the originator is vertex v, an arbitrary *tree vertex*. There is a unique path connecting v to the closest *root vertex*. We denote this path by P and the root vertex by i. The neighbor vertex of i on path P is also denoted by i_j which is the root of sub-tree T_{i_j}. Indeed, vertex v is located on T_{i_j}. Besides, imagine vertex v has k neighbors, denoted by $u_1, \cdots, u_k$, and one of these vertices is on path P and is denoted by u_i. Furthermore, we denote by T'_i vertex i and all its sub-trees rooted at $i_m, m \neq j, 1 \leq m \leq d(i)$. Clearly, $T_i = T'_i \cup T_{i_j}$. Figure 3.4 illustrates these definitions.

We first build a new Fully Connected Tree, FCT'_n, by replacing T_i by T'_i. In other words, the tree T_{i_j} is removed from FCT_n. Then, the algorithms provided in the previous section are used to broadcast in FCT'_n starting from vertex i. This gives a broadcast tree T', which are the calls made during broadcasting from vertex i in the graph FCT'_n. Now, a tree T is constructed by joining

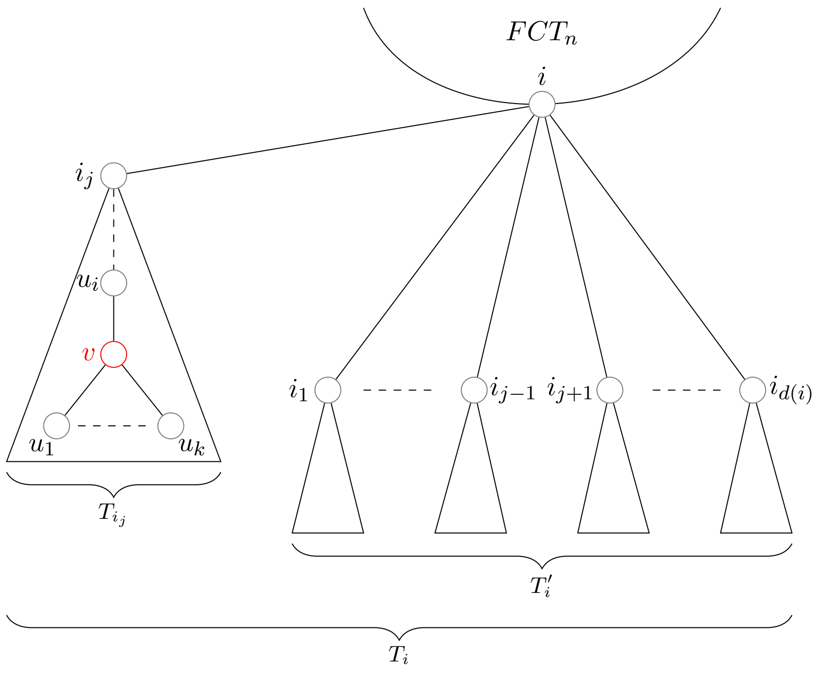


Figure 3.4: The position of vertex i and the sub-trees rooted at i. The originator, v, is located in T_{i_j}. We denote the rest of the sub-trees rooted at $i_m : m \neq j, 1 \leq m \leq d(i)$ as T'_i, in a way that $T_i = T_{i_j} \cup T'_i$.

T' and T_{i_j} using the bridge (i, i_j). The last step is to broadcast from vertex v in tree T using the well-known algorithm for broadcasting in trees (Slater et al., 1981) and output the broadcast time. The details of this algorithm are provided in Algorithm 3.

3.3.2 Proof of Correctness

Theorem 3.3.1. *Algorithm 3 generates the optimal broadcast time for a tree vertex in an arbitrary FCT_n.*

Proof. The correctness of Algorithm 3 could be proved using the following theorem from (Harutyunyan & Maraachlian, 2009b):

> *Theorem 4 of (Harutyunyan & Maraachlian, 2009b): Consider a graph H with a bridge (v_1, v_2) which divides H into two components H_1 and H_2. Let the graph H' be constructed from H such that the broadcast tree of the originator v_2 in H_2 substitutes the graph H_2. For any originator v_o in H_1 the broadcast time $B_{cl}(v_o, H) = B_{cl}(v_o, H')$.*

Algorithm 3 The broadcast algorithm $BR_T(FCT_n, v)$

```
    Input: $FCT_n = (V, E)$, originator $v$.
    Output: $B_{cl}(v, FCT_n)$
1:  $P$ = The path connecting $v$ to a root vertex in $FCT_n$;
2:  $i$ = The root vertex, $i_j$ = The neighbor of $i$ on $P$;
3:  Construct $FCT'_n = (V', E')$ as follows $V' = V \backslash V(T_{i_j})$ and $E' = E \backslash E(T_{i_j}) \backslash \{(i, i_j)\}$;
4:  Calculate $lb$ and $ub$ based on Equation (4) for $FCT'_n$;
5:  Solve $B_{Search}(FCT'_n, i, lb, ub)$;
6:  $T'$ = Broadcast tree obtained by the previous step;
7:  Construct $T = (V^T, E^T)$ as follows: $V^T = V(T') \cup V(T_{i_j})$ and $E^T = E(T') \cup E(T_{i_j}) \cup \{(i, i_j)\}$;
8:  Solve the broadcast problem for $T$ based on the well-known broadcast algorithm for trees;
9:  return $B_{cl}(v, T)$
```

variables in (Harutyunyan & Maraachlian, 2009b)	variable in Algorithm 3
v_0	v
v_1	i_j
v_2	i
H	FCT_n
H_1	T_{i_j}
H_2	FCT'_n
H'	T

Table 3.3: Mapping between the notations of Theorem 4 of (Harutyunyan & Maraachlian, 2009b) and that of Algorithm 3

The mapping between the notations in the aforementioned theorem and our algorithm is shown in Table 3.3. We only need to point out that the edge connecting vertices i and i_j is indeed a bridge since the vertices in T_{i_j} are connected to FCT_n only by this edge. Thus, removing (i_j, i) will increase the number of connected components. Based on Theorems 3.2.1 and 3.2.2, we are able to optimally broadcast within FCT'_n starting from vertex i which gives the broadcast tree T'. Joining two trees, T' and T_{i_j}, will result in another tree T for which we know the optimal broadcast scheme (Slater et al., 1981). $\square$

3.3.3 Complexity Analysis

One can notice that the problem of broadcasting from a *tree vertex* could be reduced to two sub-problems: broadcast from a *root vertex* in FCT'_n as well as broadcast within tree T. The first part takes $O(|V| \log \log n)$ using B_{Search}, while the second one could be done in $O(|V|)$ where $|V|$ is the number of nodes of the original graph FCT_n. Other steps of Algorithm 3 are also upper bounded by $O(|V|)$. Hence, the complexity of Algorithm 3 is $O(|V| \log \log n)$.

Chapter 4

A Broadcasting Heuristic for Hypercube of Trees

In this chapter, we focus on broadcasting in a useful architecture, namely the Hypercube of Trees which is constructed by attaching arbitrary trees to the vertices of a hypercube. To this aim, we propose a novel heuristic, and our numerical results show its superiority in comparison with the current upper bound for the same problem in the majority of cases.

The remainder of this chapter is structured as follows: in the following section, the problem is formally defined. In Section 4.2, we propose the heuristic when the originator is a *root* vertex. We also argue shortly why the problem will act similarly if the originator is a *tree* vertex. The performance of the heuristic is evaluated in Section 4.3 under multiple circumstances.

4.1 Preliminaries, Motivation, and Literature Review

A hypercube $H_k(V, E)$ of dimension k is a graph with 2^k vertices in which each vertex corresponds to a binary number of length k. Two vertices $u, v \in V$ are connected by an edge *iff* their binary representation differs by precisely 1 bit. Hypercubes are among the most valuable structures in network architectures since, as opposed to their small number of edges, the broadcasting could be completed within $\lceil \log n \rceil$ time units (n is the number of nodes). Due to its importance, it has been used for many real-world applications, including distributing personalized data (Ho & Johnsson, 1986) and simultaneous exchange of different packets between processors (Bertsekas, Özveren, Stamoulis, Tseng, & Tsitsiklis, 1991). Intuitively, hypercubes contain, or nearly contain, all meshes of trees, binary trees, and, more generally, every network yet discovered for parallel computing as subgraphs (Leighton, 2014). Also, they could be enhanced to some extended platforms to securely and quickly broadcast a message (Yang, Chan, & Chang, 2011). The hypercube could also be upgraded to a hypercube of trees, HT_k, in which each vertex is the root of a tree:

Definition 4.1.1. *Consider 2^k trees denoted by $T_i(V_i, E_i)$ rooted at vertex i where $0 \leq i \leq 2^k - 1$. A Hypercube of Trees $HT_k(V, E)$ is a graph in which $V = V_0 \cup \cdots \cup V_{2^k-1}$ and $E = E_0 \cup \cdots \cup E_{2^k-1} \cup E_{H_k}$ where $E_{H_k} = \{(i, j) | \text{binary representations of } i \text{ and } j \text{ differ in exactly 1 bit}\}$.*

The rationale of this design is that hypercubes benefit from a proper balance between diameter and the average degree of nodes. Ideally, an acceptable interconnection network must have a small diameter (for fast communication) and a small average of nodes' degree so that scaling up would not be impossible (Ostrouchov, 1987). Trees are another interconnection scheme that retains these vital features (Ostrouchov, 1987). So, a hypercube of trees will comply with the above-mentioned characteristics.

Definition 4.1.2. *The roots of the trees in a HT_k are called root vertices, and we call the rest of the vertices as tree vertices. A root vertex i has $d(i)$ children where each one of them is the root of a sub tree T_{i_j} where $0 \leq i \leq 2^k - 1$, $1 \leq j \leq d(i)$, $d(i) = d_i - k$, and d_i is the degree of vertex i. We also assume that $B_{cl}(i_1, T_{i_1}) \geq \cdots \geq B_{cl}(i_{d(i)}, T_{i_{d(i)}})$.*

Fig. 4.1 shows Definition 4.1.1 and 4.1.2 for $k = 3$. In this paper, we work on the broadcasting problem for a hypercube of trees. The best upper bound for this problem is presented in, (Bhabak & Harutyunyan, 2014) which is a 2-approximation algorithm, regardless of the originator. It also provides the exact broadcast time of a similar graph that contains only one tree. Their algorithm will serve as a baseline for the novel heuristic provided in this study. The numerical results illustrate that our heuristic is able to accelerate the process of broadcasting from a particular vertex in up to 90% of the experiments and achieve a speed-up of almost 30%.

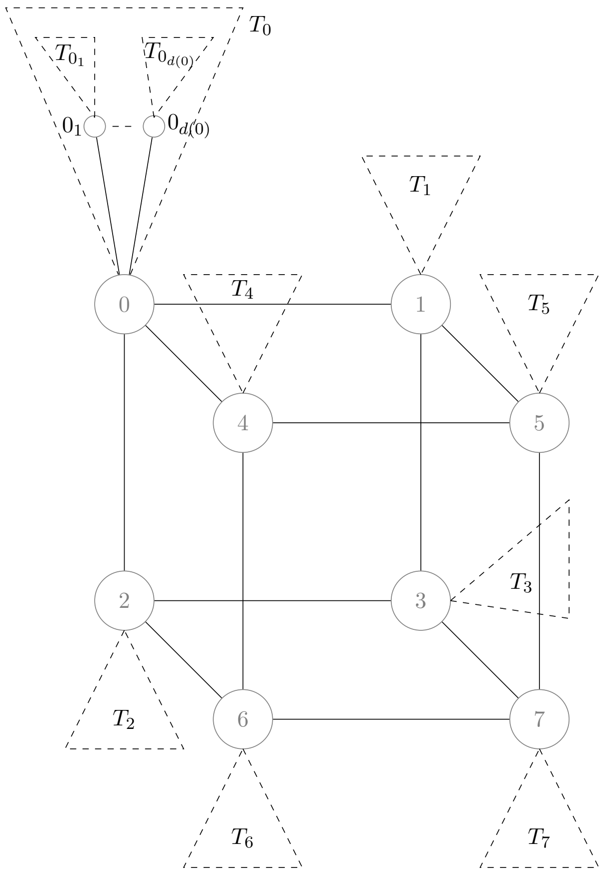


Figure 4.1: HT_3, A hypercube of trees with dimension 3

4.2 The proposed heuristic

The main idea of our heuristic is to identify the most “critical” root vertex at each time unit and to inform this vertex as soon as possible. However, there could be more than one path between an informed root vertex and that “critical” vertex in the hypercube. Thus, a greedy approach is utilized to choose the most appropriate one in which the middle vertices probably have more significant trees.

We reduce the problem of finding the broadcast time of a vertex in a given hypercube to a problem in which the aim is to study the feasibility of finishing the broadcasting from a particular vertex in τ time units. If the algorithm finds a broadcasting scheme, it will return TRUE; otherwise, the output will be FALSE. Then, we show how to tackle the original problem using a secondary algorithm.

It is assumed that the broadcast schemes for trees $T_i, 0 \leq i \leq 2^k - 1$ are available prior to the execution of the heuristic. This assumption is quite realistic because the broadcast scheme of any tree could be calculated in linear time using the algorithm provided in (Slater et al., 1981). Thus, the first step of the algorithm could be executed in parallel for all T_i's linearly. The broadcast time of a tree T_i originating from the *root* vertex i is denoted by $B_{cl}(i, T_i)$. Moreover, the algorithm discussed in (Slater et al., 1981) also gives the broadcast time of the sub-trees rooted at i_j, which will be denoted by $B_{cl}(i_j, T_{i_j})$ in the rest of this chapter.

It is easy to show that the broadcast time of a *root* vertex u in an HT_k is bounded as follows:

$$\underbrace{\max\left\{k, \max_{0\le i\le 2^k-1}\{B_{cl}(i,T_i)\}\right\}}_{lb} \le B_{cl}(u, HT_k) \le \underbrace{k + \max_{0\le i\le 2^k-1}\{B_{cl}(i,T_i)\}}_{ub} \tag{11}$$

The lower bound (lb) is trivial, and we refer the reader to the algorithm provided in (Bhabak & Harutyunyan, 2014) for the upper bound (ub). In our heuristic, we set $B_{cl}(u, HT_k) = ub$ at the first step (Algorithm 4, line 1). Then, a binary search is performed on the range of Equation (11). For each τ in this range, if the output of the heuristic is TRUE, the value of $B_{cl}(u, HT_k)$ will be updated, and the search will continue on the lower half. However, if the algorithm outputs FALSE for a given τ, the search will be performed on the upper half. The procedure of this algorithm is similar to Algorithm 1, and it is provided in Algorithm 4. The initial call for this algorithm is $B_{Search}(HT_k.i, lb, ub)$ where i is the originator and lb and ub come from Equation (11).

In the rest of this section, the heuristic used in Algorithm 4 will be discussed, which could be applied to the *root* vertices, and in order to do so, some notations and definitions must be discussed first.

Notation 4.2.1. *The set of informed root vertices of HT_k is denoted by V_I, while V_U is the set of uninformed root vertices. We denote the set of root vertices by V_{H_k} where $V_{H_k} = V_I \cup V_U$.*

Notation 4.2.2. *For $j \in V_U$, rem_j is the remaining time that other informed root vertices can avoid j and take care of any other uninformed root vertex, and it is calculated as follows: $rem_j = \tau - t - B_{cl}(j, T_j) - dist(j, V_I)$ in which τ is the candidate broadcast time, t is current time, and $dist(j, V_I)$ is the length of the shortest path from j to the closest informed root vertex. Moreover, for $j \in V_I$, we set $rem_j = NULL$.*

Algorithm 4 The modified Binary Search algorithm $B_{Search}(HT_k, i, lb, ub)$

```
    Input: $HT_k = (V, E)$, originator $i$, lower bound $lb$, and the upper bound $ub$.
    Output: An improved broadcast time for $B_{cl}(i, HT_k)$ denoted by $b$.
 1: $b = ub$;
 2: if $lb > ub$ then
 3:     return $ub$
 4: end if
 5: $mid = lb + \lfloor \frac{ub-lb}{2} \rfloor$;
 6: if $br(HT_k, i, mid)$ then
 7:     update $b = mid$;
 8:     return $B_{Search}(HT_k, i, lb, mid - 1)$
 9: else
10:     return $B_{Search}(HT_k, i, mid + 1, ub)$
11: end if
12: return $b$
```

The rationale for calculating rem_j is straightforward; the total remaining time is $\tau - t$. Furthermore, vertex j needs $B_{cl}(j, T_j)$ time units to complete the broadcasting within T_j once informed. Besides, it takes at least $dist(j, V_I)$ time units for vertex j to be informed by any informed *root* vertex. It should be noted that the distance of two vertices in a hypercube could be calculated in constant time by counting the number of different bits in their corresponding binary numbers.

Definition 4.2.1. *For a path P in a hypercube connecting two root vertices i and j, the overall deadline of the path is defined as follows:*

$$d_P = \sum_{m \in P} rem_m \tag{12}$$

Definition 4.2.2. *For a path P in a hypercube connecting two root vertices i and j, the critical deadline of the path is defined as follows:*

$$c_P = \min_{m \in P} \{rem_m\} \tag{13}$$

By definition, among all paths existing between two root vertices i and j, the smaller c_P and d_P for a path P, the higher the priority of the P, since the middle vertices lying on P have less remaining time until they must be considered by informed vertices. These definitions will serve as a heuristic for the proposed algorithm when the algorithm is obligated to choose among several paths. We denote a path P chosen by our heuristic by *selected path*.

Definition 4.2.3. *A root vertex i is covered if either $i \in V_I$ or i is located on a selected path: $covered(i) = TRUE$. Otherwise, $covered(i) = FALSE$.*

Definition 4.2.4. *A path P is said to be valid if none of the vertices lying on P are covered, that is, the validity of a path P (when $|P| > 2$) connecting vertex i to vertex k, and then, vertex k to vertex j (with probably multiple steps) is calculated as follows:*

$$valid(\underset{i \to k \to \cdots \to j}{P}) = \neg covered(k) \wedge \cdots \wedge \neg covered(j) \tag{14}$$

Additionally, the validity of any path with a length of two is equal to the negation of the destination's covered value.

The main idea of the algorithm is as follows: for an informed *root* vertex i, there are two possible scenarios; whether this vertex has enough time to contribute to the hypercube and inform other uninformed *root* vertices, or it must start the broadcasting within its tree T_i. The decision for these two options is made by comparing the current value of $B_{cl}(i, T_i)$ and the remaining time $\tau - t$.

When $B_{cl}(i, T_i) = \tau - t$, the root vertex i has to start broadcasting within its tree; thus, the provided heuristic will follow the algorithm given in (Slater et al., 1981) for broadcasting in a tree optimally. On the other hand, when $B_{cl}(i, T_i) < \tau - t$, the root vertex i has at least 1 time unit left to continue broadcasting in the hypercube. Two questions must be answered in such a case:

(1) How i chooses a root vertex j to send the message toward that vertex?

(2) Which path should i take toward j?

For the first question, we select a vertex j with the minimum value of rem_j among the members of the following set:

$$V_U \backslash \{m | covered(m) = True\} \tag{15}$$

Among the uninformed root vertices, the ones already covered by other vertices are not considered a good choice because, although they are not informed at the current time, another root vertex is already on its way to inform them. Hence, they cannot be selected again by another root vertex. For selecting the destination, a root vertex with the minimum value of rem_j is selected from the set in Equation (15). It has less time to the moment that it must be informed compared to other candidate vertices. Also, ties are broken randomly.

To answer the second question, we deployed the following heuristic: among all *valid* paths between i and j with the minimum value of c_P, choose the one with the minimum value of d_P. Again, ties are broken randomly. The rationale of our heuristic is that a path P is a good candidate if the most "critical" uncovered root vertex, which has the smallest value of c_P is located on it. This "critical" vertex must be informed as quickly as possible; otherwise, the algorithm will fail. However, that critical vertex could lie on many *valid* paths. Among those, the one that has the smallest value of d_P is desirable since all the middle vertices on P are relatively critical.

Once a path P is *selected*, the source (i) knows how the message should be sent toward the target (j). Moreover, all the vertices on the path know their next call once informed.

Definition 4.2.5. *A root vertex is called busy if it is located on a selected path. An informed-busy-root vertex will follow the path that it is supposed to go along with.*

Definition 4.2.6. *An informed root vertex is called stuck if all its neighbors in the hypercube are either covered or informed.*

A pseudo-code for the heuristic is provided in Algorithm 5. As can be seen, important decisions are to be made when $B_{cl}(i, T_i) < \tau - t$ for an informed root vertex. In this case, the root vertex is checked to be *busy* or *stuck*. If i is *busy*, it lies on a selected path and must perform the call it is supposed to obey (Line 9 of Algorithm 5). Besides, if the root vertex is *stuck*, there is nothing it can do to accelerate the process (Line 13 of Algorithm 5). However, if none of those situations are valid for i, it must find another uninformed root vertex j from the set of Equation (15). But, there may be no *valid* path from vertex i to the vertex with the minimum value of rem_j. Thus, vertex i cannot be responsible for that vertex (Lines 15-16 of Algorithm 5).

Once a destination is chosen, among all valid paths leading to vertex j, one is selected (P) based on the heuristic (Lines 17-19 of Algorithm 5). Now, not only does vertex i know its next call, but all the vertices lying on P are also aware of their call once they get informed. Therefore, they will be

Algorithm 5 The broadcast algorithm $BR_\tau(HT_k, i, \tau)$

Input: $HT_k = (V, E)$, originator i, candidate broadcast time τ.

Output: FALSE if τ cannot be the broadcast time, TRUE if broadcasting can be accomplished in at most τ time units.

```
1: **Initialize** $V_I = \{i\}$, $V_U = V_{H_k} - V_I$, $covered(u \in V_U) = FALSE$, $covered(i) = TRUE$, $rem_i = NULL$;
2: **for each** $t$ such that $0 \leq t \leq \tau - 1$ **do**
3:     **for each** $u \in V_U$ **do**
4:         **update** $rem_u = \tau - t - B_{cl}(u, T_u) - dist(u, V_I)$
5:     **end for**
6:     **for each** $i \in V_I$ **do**
7:         **if** $B_{cl}(i, T_i) < \tau - t$ **then**
8:             **if** $busy(i)$ **then**
9:                 $i$ informs $v$ following its path;$V_I = V_I \cup \{v\}$, $V_U = V_U - \{v\}$;
10:                **continue**;
11:            **end if**
12:            **if** $i$ is stuck **then**
13:                **continue**;
14:            **end if**
15:            Choose $j$ with the minimum value of $rem_j$ from the set of Equation (15) in a way
16:            that there is at least one *valid* path from $i$ to $j$;
17:            Denote the valid paths by $VP = \{p_1, \cdots, p_r\}$;
18:            $MC = \{p \in VP | c_p = \min_{p' \in VP}\{c_{p'}\}\}$;
19:            $selected = \{p \in MC | d_p = \min_{p' \in MC}\{d_{p'}\}\}$;
20:            **if** $|selected| > 1$ **then**
21:                Select one randomly;
22:            **end if**
23:            For all mid-vertices on the selected path, set $covered(mid - ver) = TRUE$,
24:            $busy(mid - ver) = TRUE$;
25:            $i$ informs $v$ which is the first vertex on the selected path;
26:            $V_I = V_I \cup \{v\}, V_U = V_U - \{v\}$;
27:        **else**
28:            **if** $B_{cl}(i, T_i) = \tau - t$ **then**
29:                Follow the broadcast scheme in trees;
30:                **update** $B_{cl}(i, T_i)$ with regard to a sub tree $T_{i_j}$ which has been informed;
31:            **else**
32:                **return** FALSE;
33:            **end if**
34:        **end if**
35:    **end for**
36: **end for**
37: **return** TRUE;
```

labeled as covered and busy vertices (lines 23-24). Lastly, vertex i performs its call (line 25), and the newly informed root vertex will be added to V_I (line 26).

In cases when the remaining time equals the broadcast time of a root vertex i within its tree, vertex i has no choice but to perform at least one call in its biggest subtree. Possibly, its broadcast time could decrease substantially by only one call. It happens when there is a huge subtree (such as T_{i_1}), while other subtrees are relatively small. Therefore, making one call could free vertex i for a few time units. This procedure will be done when $B_{cl}(r_i, T_i)$ is updated (lines 28-30). Lastly, if the remaining time is smaller than the time needed for vertex i to finish in T_i, the algorithm returns FALSE since there is a vertex that cannot finish the broadcasting even in its tree (line 32).

Lastly, we shortly argue that the problem will remain almost the same when the originator is not a *root* vertex. Assume the originator is vertex u, located in the subtree T_{i_j}. Thus, u also belongs to the tree T_i, and the closest root vertex to u is i. Finding $B_{cl}(u, HT_k)$ could be divided into two simpler subproblems. First, we calculate $B_{cl}(i, HT'_k)$ using the proposed heuristic where HT'_k is the same graph with HT_k, only the tree T_{i_j} is removed. This gives a broadcast tree rooted at vertex i in HT'_k. Denote this tree by T.

Secondly, using Theorem 4 of (Harutyunyan & Maraachlian, 2009b), one can argue that $B_{cl}(u, HT_k)$ could be calculated easily by replacing HT'_k with the tree obtained from the first step (or T). In other words, we merge T with T_{i_j} and find the broadcast time of u in the resulting tree. Therefore, the problem will be reduced to a more straightforward problem of finding the broadcast time of a vertex in a tree which is solvable using the algorithm in (Slater et al., 1981).

4.3 Evaluation

In this section, the performance of the proposed heuristic is examined under multiple circumstances. Also, it should be noted that the performance of our heuristic is upper bounded by that of (Bhabak & Harutyunyan, 2014), which is a 2-approximation since we never find a broadcast time worse than ub (in Equation (11)). However, the numerical results illustrate the superiority of our heuristic in most cases by a wide margin.

4.3.1 Experimental setup

To evaluate the performance of the heuristic, we compare our results with the algorithm provided in (Bhabak & Harutyunyan, 2014). In order to do so, in each experiment, a hypercube is generated with dimension k. Then, a random procedure is followed for attaching trees to root vertices. Once the HT_k is constructed, the performances of both algorithms are compared in terms of the time units required to finish the broadcasting from a randomly chosen originator. The numerical results are averaged over 1000 executions for each k. Furthermore, all simulations of this study are performed on an Intel Core i7 CPU with a 2.7 GHz frequency. We also used Python for the simulation due to its reputation and also great speed and simplicity. Generating numerical results for small values of k takes a few seconds, while it needs up to a minute for larger graphs.

For generating trees, random samples from a Gaussian distribution (Patel & Read, 1996) are drawn. A random sample will then be rounded to an integer. We have used the randomly generated integer for two purposes: Firstly, a random integer is generated, illustrating the number of children a particular root vertex i has. Also, for each child, i_j, another random number is generated, which demonstrates the broadcast time of the sub-tree T_{i_j}, or $B_{cl}(i_j, T_{i_j})$. The probability density function for the Gaussian distribution is as follows (Patel & Read, 1996):

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{16}$$

μ is the mean, and σ is the standard derivation. By definition, this function is more likely to return samples lying close to the mean rather than those far away (See Fig. 4.2). We convert a negative number generated by a Gaussian function to zero. Once the random trees are generated, the broadcast time of the root vertex i is calculated using the equation provided in (Slater et al., 1981):

$$B_{cl}(i, T_i) = \max_{1 \leq j \leq d(i)} \{j + B_{cl}(i_j, T_{i_j})\} \tag{17}$$

We report the average number of vertices of HT_k for each k as a prominent element of the performance of our heuristics. Our numerical results reveal that a HT_k could have millions of

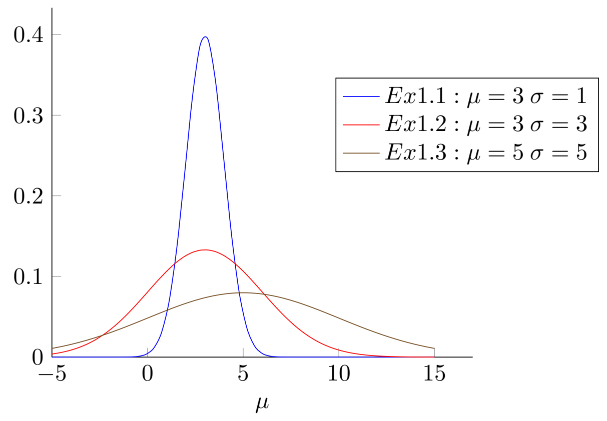


Figure 4.2: Three Gaussian distribution curves

vertices in cases where k is a small number, such as 8. It should be noted that having more than a million nodes in random graphs is quite natural. Note that a subtree with a broadcast time of t could have up to 2^t vertices if it has a form of a binomial tree of size t.

We measure performance improvement as follows:

$$\text{gain} = \frac{B_{cl_{[2-app]}}(i, T_i) - B_{cl}(i, T_i)}{B_{cl_{[2-app]}}(i, T_i)} \tag{18}$$

In which $B_{cl_{[2-app]}}(i, T_i)$ is the 2-approximation broadcast time of the algorithm provided in (Bhabak & Harutyunyan, 2014), and the other term is the broadcast time calculated by our heuristic. The average gain is the value of Equation (18) that is averaged over all experiments for a particular k. Lastly, the *success rate* will be reported as the fraction of the experiments in which $gain > 0$, or in other words, the cases when our heuristic outperforms that of (Bhabak & Harutyunyan, 2014):

$$\text{success rate} = \frac{\text{number of cases where } B_{cl}(r_i, T_i) < B_{cl_{[2-app]}}(r_i, T_i)}{\text{total number of experiments}} \tag{19}$$

4.3.2 Numerical results

We performed two experiments to study the performance of the proposed heuristic that are discussed hereafter.

Experiment 1

The performance of the heuristic is evaluated concerning the size of the hypercube (k) under three distinct circumstances (See Fig. 4.2). For Ex1.1, $\mu = 3$ and $\sigma = 1$, meaning that each root vertex is more likely to have a number of children that lies in the interval of $[2, 4]$, each of which has a random broadcast time that follows the same distribution. In other words, the broadcast time of root vertices will be very similar in most cases. The results of this experiment are shown in Table 4.1. As k grows, the performance of the heuristic drops, leading to almost the same performance as (Bhabak & Harutyunyan, 2014) when $k = 7, 8$. We will discuss the reason behind this phenomenon soon. However, It seems that our heuristic is able to outperform its counterpart when the value of k is not high. For example, outperforming (Bhabak & Harutyunyan, 2014) in 30% of the experiments when $k = 3$ leads to an average speedup of almost 7.5%.

In Ex1.2, μ and σ are both set to 3, and the interval is $[0, 6]$. In other words, each root vertex will probably have 0-6 subtrees, each having a broadcast time of 0-6. Table 4.2 illustrates the results of Ex 1.2. Although the size of HT_k's is greater than that of Ex1.1, the algorithm's performance is much better since a *success rate* of almost 55-75% is observed, leading to a speed-up of 10-25%.

In the last experiment, we set $\mu = 5$, and $\sigma = 5$ having an interval of $[0, 10]$. Hence, in the last experiment, the root vertices are more likely to have different broadcast times within their trees. Table 4.3 gives the performance of our heuristic in Ex1.3. A solid performance, even in large graphs, is observed for our heuristic in this case, where in almost 90% of the experiments, we can outperform the current upper bound for this problem, leading to a 25-30% speed-up on average.

Comparing the results of Ex1.1, Ex1.2, and Ex1.3, we first argue that the performance of our

Table 4.1: Numerical Results for Ex1.1: $\mu = 3, \sigma = 1$

k	**average $\|V\|$**	**success rate**	**average gain**
3	115.86	29.79%	7.55%
4	233.55	28.49%	5.84%
5	466.09	21.3%	3.71%
6	932.49	5.7%	0.85%
7	1862.98	1.3%	0.16%
8	3729.59	0.2%	0.02%

Table 4.2: Numerical Results for Ex1.2: $\mu = 3, \sigma = 3$

k	average $\|V\|$	success rate	average gain
3	614.28	73.5%	24.45%
4	1189.54	78.6%	23.31%
5	2481.77	74.5%	19.81%
6	5008.22	69.2%	15.55%
7	10003.63	63.2%	12.5%
8	19927.99	54.7%	9.76%

heuristic is independent of the value of k, and subsequently, $|V|$. It is also claimed that our heuristic will be genuinely effective when the value of σ is high, regardless of the value of μ. To test this claim, we designed the following experiment.

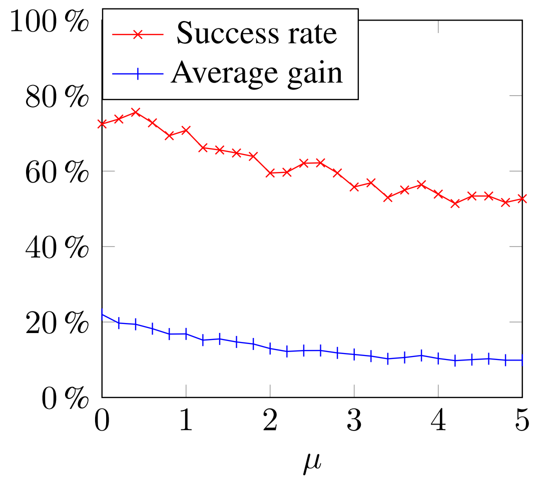
Experiment 2

To test the effect of μ and σ, two more experiments have been conducted in which the value of k is set to 5. In Ex2.1, σ is set to 2, but μ increases gradually from 0 to 5, with a 0.2 step. The aim of Ex2.2, on the other hand, is to fix μ to 2, while σ grows from 0 to 5, gently. More clearly, in Ex2.1, the size of HT_k grows by attaching larger trees to each root vertex, but $B_{cl}(i, T_i)$ is likely to be very close for all root vertices. However, Ex2.2 invests in situations where the margin among the broadcast times of root vertices widens. The results of these experiments in terms of success rate and the average gain are represented in Figs. 4.3a and 4.3b, respectively.

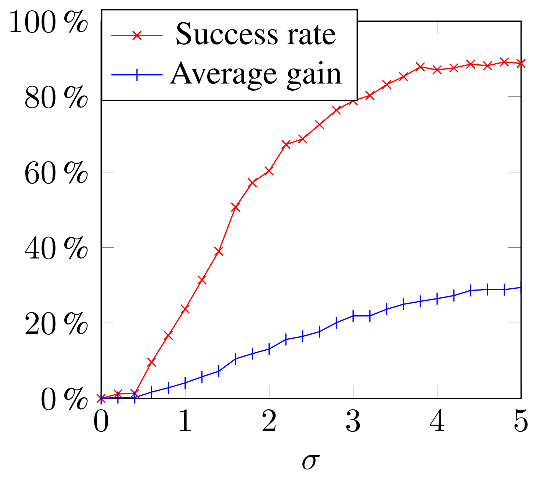
As far as Fig. 4.3a is concerned, the Success rate experiences a gradual decline when the value of μ increases, but it could still beat the best upper bound in at least 50% of the experiments,

Table 4.3: Numerical Results for Ex1.3: $\mu = 5, \sigma = 5$

k	average $\|V\|$	success rate	average gain
3	185255.48	81.20%	28.65%
4	280823.85	86.00%	28.82%
5	704372.23	88.10%	27.35%
6	1313690.28	89.30%	26.70%
7	3532669.06	90.50%	25.85%
8	5245921.21	90.10%	24.10%



(a) Ex2.1: $k = 5, \sigma = 2$



(b) Ex2.2: $k = 5, \mu = 2$

Figure 4.3: The performance of the algorithm when the value of μ or σ increases

having gained up to 20% in comparison with ub. It shows that the proposed algorithm can perform impressively when the value of μ grows, and as an immediate consequence, the size of the graph multiplies. Accordingly, we argue that the performance of the proposed heuristic could be irrelevant to the graph's size to a certain extent.

Additionally, as seen in Fig. 4.3b, the proposed heuristic can achieve a remarkable quality when the value of σ increases. Hence, the broadcast time of trees or $B_{cl}(i, T_i)$ are more likely to differ in comparison with the cases when σ is close to 0, and thus, $B_{cl}(i, T_i)$ is probably the same for all root vertices. This is because in the first case, HT_k will likely have a huge tree rooted at vertex j, many medium-size ones, and some small trees. Unlike the current upper bound, our heuristic can successfully detect this situation and act in a way that the originator is obligated to choose vertex j and send the message toward this vertex to enable it to initiate the broadcasting within its tree as soon as possible. In the meantime, other root vertices will be informed one by one. Still, since their broadcast time probably differs from that of j, there is no harm in informing them with a delay of 1 or 2 time units, as our numerical results also support this idea and show at least an 80% success rate when $\sigma \geq 3$.

On the other hand, when σ is small, all root vertices will have very close broadcast time. In this case, one may easily argue that the optimal decision is quite similar to the current upper bound: to inform the root vertices in k time units and then finish the broadcasting in trees. The reason is that when a root vertex starts to inform its tree at time $t < k$, it will not accelerate the whole process because, in the long run, the bottleneck would be the last root vertex that has been informed, and the optimal decision is to avoid delaying the process of informing the root vertices. Therefore, there is not much gap for our heuristic to work with, and it is no surprise that it cannot outperform the current upper bound in this case. This will also justify the results of Ex1.1, where the proposed heuristic is not successful, especially when k grows (note that the value of σ is very small in that experiment). All in all, our results illustrate that the heuristic proposed in this study can outperform the current upper bound for the problem of broadcasting in a hypercube of trees in many cases. Also, when the gap between the upper bound and the optimal solution shrinks, our heuristic will perform as adequately as the best-known upper bound.

Chapter 5

Fully-adaptive Model for Broadcasting with Universal Lists

An ordered list of neighbors is assigned to each node via a broadcasting algorithm, which specifies the order in which messages are transmitted to neighbors at each node. The list assigned to each node in classical broadcasting varies depending on the source. Therefore, each node must have a large local memory to keep the lists corresponding to various sources and be aware of the source to select the appropriate list to perform broadcasting from each potential source. Because each message should carry the name of its source of origin, this increases the number of message bits sent throughout the network and necessitates significant local memory at each node (Diks & Pelc, 1996; J.-H. Kim & Chwa, 2005).

Another variant of broadcasting is introduced in the literature to handle the above-mentioned drawbacks. A vertex v of a network is given a $universal\ list$, denoted by l_v. When a vertex receives the message, it should transmit it to its neighbors with respect to the fixed ordering given in the list, regardless of the originator. Two sub-models are defined using universal lists:

- *Non-adaptive:* Once a vertex v receives the message, it will re-transmit it to all the vertices on l_v, even if v has received it from that particular vertex. The broadcast time of a graph G following this model is denoted by $B_{na}(G)$.

  We say a call from v to u at time t is *unnecessary* if u is informed at any time $t' < t$. The definition of the non-adaptive model implies many *unnecessary calls* in which the message is returned to the sender. This would make this model the slowest among all while having the best space complexity since the list is the only thing that should be maintained.

- *Adaptive:* Once informed, a vertex v will send the message to its neighbors according to l_v, but it will skip the neighbors from which it has received the message. The broadcast time of a graph G following this model is denoted by $B_a(G)$.

  Although the number of *unnecessary call*s drops in the adaptive model compared to the non-adaptive model, many calls might still be unnecessary. For instance, a vertex v may transmit the message to one of its neighbors u, which is already informed by its other neighbors. Each vertex also needs extra storage to keep track of the nodes from which the message has been received. This will accelerate the broadcasting process under this model compared to the non-adaptive model but will increase memory usage.

In this chapter, we propose the third sub-model for the problem of broadcasting with universal lists, which is faster than adaptive and uses less memory than the classical model. To this aim, in Section 5.1 we present the details of the newly introduced Fully-adaptive model. Later, the broadcast time of several graphs under this model is given in Section 5.2.

5.1 Fully-adaptive Model

To achieve a reasonable balance between the speed of broadcasting and space complexity, we define the Fully Adaptive model, another sub-model for broadcasting with universal lists. This model is faster than adaptive and more efficient than the classical model in terms of space complexity. In the fully-adaptive model, each vertex $v \in V(G)$ is equipped with a list l_v. Denote the degree of vertex v by d_v. The size of l_v is at most the number of neighbors of v: $|l_v| \leq d_v$. Once vertex v is informed, it will follow its list and pass the message to the first vertex on l_v, which is not already informed. In other words, not only does the sender skip all those neighbors from which it received the message, but it should also skip all other informed vertices. Thus, as in the classical model, no unnecessary call is made following the fully-adaptive model. Following this model, we denote the broadcast time of a graph G by $B_{fa}(G)$.

Theorem 5.1.1. *$B_{cl}(G) \leq B_{fa}(G) \leq B_a(G) \leq B_{na}(G)$, for any connected graph G.*

Proof. We only need to prove $B_{cl}(G) \leq B_{fa}(G)$ and $B_{fa}(G) \leq B_a(G)$. The correctness of the last inequality is established in (Diks & Pelc, 1996).

(1) $B_{cl}(G) \leq B_{fa}(G)$: Note that a broadcast scheme for any variant of broadcasting could be viewed as a spanning tree T, rooted at the originator. An edge $(u, v) \in T$ corresponds to time t when u made a call to inform v, or vice versa. By definition, any spanning tree under the fully-adaptive model is a valid spanning tree and hence, a broadcast scheme for the classical model. Thus, $B_{cl}(G) \leq B_{fa}(G)$.

(2) $B_{fa}(G) \leq B_a(G)$: Consider any broadcast scheme for an arbitrary graph G under the adaptive model. Under the fully-adaptive model using the same list, in addition to the vertices that a particular vertex has received the message from, it will also skip all other vertices that are already informed, thus: $B_{fa}(G) \leq B_a(G)$.

□

Later in Theorem 5.2.1, we will prove that for any tree T, $B_{cl}(T) = B_{fa}(T) = B_a(T)$. So, the bounds presented in Theorem 5.1.1 are tight and cannot be improved.

Following Theorem 5.1.1, Table 5.1 compares all 4 models. In the fully-adaptive model, a node only requires local knowledge over its neighbors and not the whole network as it does in the classical model. Moreover, a universal list is maintained throughout the process, requiring less space than keeping a list per originator, which is the case in the classical model. Therefore, the fully-adaptive model is more efficient than the classical model regarding space complexity. Moreover, in the adaptive model, a vertex has to know the neighbors from which it has received the message, while in the non-adaptive model, even this information is not kept in the memory. Thus, the non-adaptive model is the most efficient in terms of space complexity. However, considering the speed of broadcasting and by looking at Theorem 5.1.1, it is clear that the non-adaptive model is the slowest among all. In summary, one may look at the fully-adaptive model as a model that tries to behave similarly to the classical model using a universal list.

In terms of space complexity, in the non-adaptive model, the list is the only data structure that should be maintained that needs at most $\sum_{1\leq i\leq n} d_i$ space. In the adaptive model, in addition to the

Table 5.1: The Comparison Among Different Variants of Broadcasting

No.	Model	Symbol	In a call $u \to v$, the receiver ...	No. of unnecessary calls	Required Space	Speed
1	Non-adaptive	$B_{na}(G)$	Could be any neighbor	Many	$\sum_{1\leq i\leq n} d_i$	Very Slow
2	Adaptive	$B_a(G)$	Is a neighbor which hasn't sent to u yet	Few	$2 \times \sum_{1\leq i\leq n} d_i$	Slow
3	Fully Adaptive	$B_{fa}(G)$	Is always uninformed	0	$2 \times \sum_{1\leq i\leq n} d_i$	Moderate
4	Classical	$B_{cl}(G)$	Is always uninformed	0	$n \times \sum_{1\leq i\leq n} d_i$	Very Fast

above lists, each vertex v_i should maintain a set of size d_i in order to keep track of the vertices that have sent the message to vertex v_i. Therefore, the required space is at most $2 \times \sum_{1\leq i\leq n} d_i$. This is the same for the fully-adaptive model, but those sets store the informed neighbors, which should be updated after each time unit. On the other hand, in the classical model, firstly, those lists differ according to every possible originator. Thus, the required space is $n \times \sum_{1\leq i\leq n} d_i$. Secondly, the message bits increase since the identity of the originator should be carried on with the message. In summary, in universal lists models, the space complexity is $O(|E|)$, while in the classical model, it requires $O(|V| \cdot |E|)$. Hence, the choice of a suitable model heavily depends on the available resources and the requirements of the network.

Considering a graph $G = (V, E)$, a broadcast scheme for the non-adaptive, adaptive, or fully-adaptive model can be viewed as a matrix $\sigma_{n\times\Delta}$, where row i of σ corresponds to an ordering of the neighbors of vertex v_i. Assuming this vertex has degree d_i, the cells $\sigma_{[i][d_i+1]}, \sigma_{[i][d_i+2]}, \cdots, \sigma_{[i][\Delta]}$ will be NULL. By definition: $\Delta = \max\{d_i : 1 \leq i \leq n\}$. We denote the set of all possible schemes by Σ. In this study, we assume that row i of σ contains all neighbors of v_i in some order unless mentioned otherwise.

Let M be one of the three models using universal lists ($M \in \{na, a, fa\}$) and fix a graph G. For any broadcast scheme $\sigma \in \Sigma$, we denote by $B_M^\sigma(v, G)$ the time steps needed to inform all the vertices in G from the source v while following the scheme σ under model M. Moreover, the broadcast time of a graph G under model M with scheme σ is defined as the maximum $B_M^\sigma(v, G)$ over all possible originators $v \in V(G)$: $B_M^\sigma(G) = \max_{v\in V}\{B_M^\sigma(v, G)\}$. Lastly, $B_M(G)$ is the minimum $B_M^\sigma(G)$ over all possible schemes $\sigma \in \Sigma$: $B_M(G) = \min_{\sigma\in\Sigma}\{B_M^\sigma(G)\}$.

For instance, Table 5.2 shows a broadcast scheme σ for the graph given in Figure 5.1-a. Consider vertex v_1 as the originator. The broadcast scheme of the graph is shown in Figures 5.1-b, 5.1-c, and 5.1-d for fully-adaptive, adaptive, and non-adaptive models, respectively.

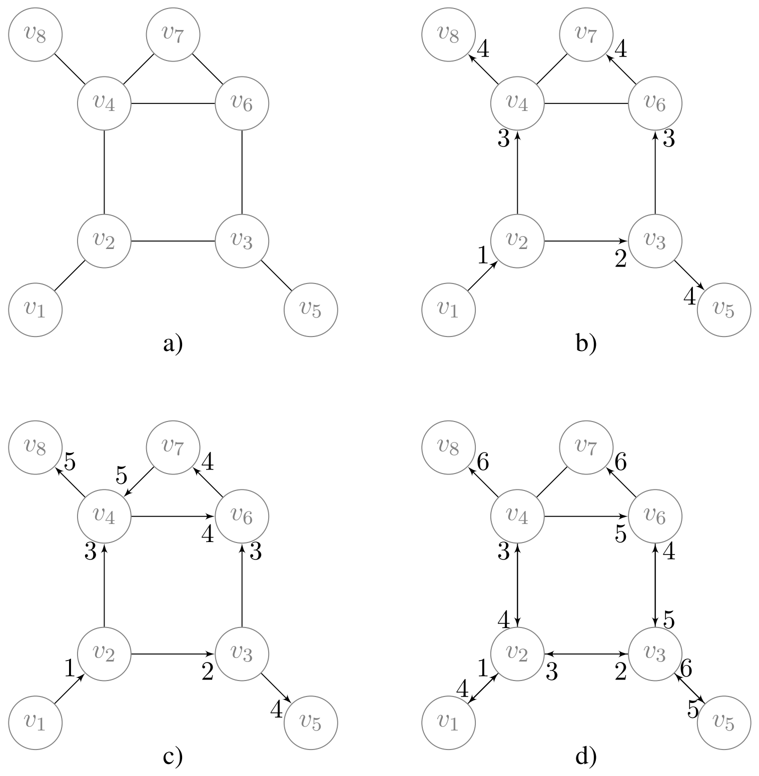


Figure 5.1: a) A Graph G, b) Broadcast scheme of G under fully-adaptive model: $B^{\sigma}_{fa}(v_1, G) = 4$, c) Broadcast scheme of G under adaptive model: $B^{\sigma}_{a}(v_1, G) = 5$, d) Broadcast scheme of G under non-adaptive model: $B^{\sigma}_{na}(v_1, G) = 6$

5.1.1 Assumptions, Architecture, and Applications

Now we will present some insights regarding the implementation of the fully-adaptive model in real-world networks:

Assumptions: Consider a non-faulty network in which the links among entities are established. The originator could be regarded as a server that tries to distribute a unique and heavy-weight message to all clients. Also, clients will relay the message and send it to each other. Further suppose the message consists of a header and a payload. The header only contains critical information, such as SYN and ACK, in the 3-way handshake procedure of TCP protocol (Ohsita, Ata, & Murata, 2004), while the payload contains the actual data.

Architecture: In the described situation, it makes sense to avoid unnecessary calls from a sender to an informed receiver. Thus, both adaptive and non-adaptive models result in a notable delay. Also, considering a massive network, equipping entities with unlimited memory seems to be unrealistic

(Vissicchio, Vanbever, Cittadini, Xie, & Bonaventure, 2017), as it is the case in the classical model. Consequently, the proposed fully-adaptive model could be deployed, but the network architecture must allow the members to be aware of their neighbors' states at any given time. To this aim, two approaches may be used: Push model or Pull model. In the first one, once a member gets informed, it will update its state for all its neighbors via a lightweight message. However, in the latter approach, once an entity wants to send a message to one of its neighbors, it will send a special request to know the state of its neighbor. The neighbor will respond as either informed or uninformed. Both messages are lightweight. Afterward, the actual heavy data, a.k.a payload, should be transmitted. Even though both architectures could be used for the proposed model, the Push model fits it better since it will only result in a single additional and lightweight message per neighbor. In contrast, in the Pull model, each neighbor will be asked for an update which is accompanied by a response.

Applications: As a potential application of the proposed model, we refer to the network update procedure in Software Defined Networks (SDNs). In order to keep the network in an efficient and working state, the network operator has to perform several tasks, such as changing routing policies, adjusting links' weights, and modifying traffic engineering schemes (D. Li, Wang, Zhu, & Xia, 2017). These heavy tasks, usually known as "network updates", are highly influential on the performance of the networks and could affect the quality of service (QoS) (D. Li et al., 2017; Sezer et al., 2013). Besides, the desire for user-controlled management of forwarding in network nodes led to the emergence of SDNs (Sezer et al., 2013). The separation of control and data plane in SDNs has enabled researchers to suggest new solutions to network update problems. In particular, the data plane only forwards packets, while the routing and load balancing decisions for SDN switches are

Table 5.2: An ordering of the vertices of the graph presented in Figure 5.1,a

Sender	Ordering of receivers			
v_1	v_2	Null	Null	Null
v_2	v_3	v_4	v_1	Null
v_3	v_2	v_6	v_5	Null
v_4	v_2	v_6	v_8	v_7
v_5	v_3	Null	Null	Null
v_6	v_3	v_7	v_4	Null
v_7	v_6	v_4	Null	Null
v_8	v_4	Null	Null	Null

made in a logically centralized controller (Scott-Hayward, O'Callaghan, & Sezer, 2013). However, as mentioned in (D. Li et al., 2017), SDN switches forward packets based on their own forwarding tables, and operators need to design and optimize the update mechanisms carefully. This study focuses on this particular application. As a network manager, we propose optimal or near-optimal broadcast schemes for some common interconnection networks, such as Trees, Grids, Tori, and Cube Connected Cycles, under the fully-adaptive model.

5.2 Results on the Fully-adaptive Model

This section presents optimal broadcast schemes for trees, grids, and cube-connected cycles under the fully-adaptive model. We also provide an upper bound for tori. These results show that despite a significant reduction in memory usage in the fully-adaptive model compared to the classical model, $B_{cl}(G) = B_{fa}(G)$, or their values are very close.

5.2.1 Trees T

We begin by proving a general statement for trees.

Theorem 5.2.1. *For any tree T, $B_{cl}(T) = B_{fa}(T) = B_a(T)$.*

Proof. From Theorem 5.1.1, observe that $B_{cl}(T) \leq B_{fa}(T) \leq B_a(T)$. Moreover, as mentioned in (Slater et al., 1981) and (Diks & Pelc, 1996), $B_{cl}(T) = B_a(T)$ for any tree T. The result follows. Note that an optimal broadcast scheme for a tree T under the classical model (Slater et al., 1981) will be a valid and optimal broadcast scheme for the same tree under the fully-adaptive model. $\square$

5.2.2 Grids $G_{m\times n}$

It is proved that $B_{cl}(G_{m\times n}) = B_a(G_{m\times n}) = m + n - 2$ (Diks & Pelc, 1996; Farley & Hedetniemi, 1978). Based on Theorem 5.1.1, we infer the following:

Corollary 5.2.1. $B_{fa}(G_{m\times n}) = m + n - 2$.

We need to point out that the broadcast scheme presented in (Diks & Pelc, 1996) for Grids under the adaptive model will also be a valid broadcast scheme under the fully-adaptive model. However, we will give another more straightforward and optimal broadcast scheme σ under the fully-adaptive model. The ordering of a vertex (x, y) in σ is denoted by $l_{(x,y)}$, and it consists of (at most) its four neighbors in the following order:

(1) The above neighbor or $(x, y + 1)$,

(2) The below neighbor or $(x, y - 1)$,

(3) The right neighbor or $(x + 1, y)$,

(4) The left neighbor or $(x - 1, y)$,

Consider an arbitrary originator (x, y) in a grid $G_{m\times n}$. This vertex will start broadcasting within its column, and this will be finished in at most $n - 1$ time units following σ. Note that every vertex will prioritize its above and below neighbors first, and the message will be broadcast in column x. Now that all the vertices in column x are informed, they will simultaneously pass the message to the vertices on their rows. This process will not take more than $m - 1$ time units. Therefore, for an arbitrary originator $\forall x, y; 1 \leq x \leq m, 1 \leq y \leq n$: $B^{\sigma}_{fa}\big((x, y), G_{m\times n}\big) \leq m + n - 2$.

5.2.3 Tori $T_{m\times n}$

In what follows, we will study $B_{fa}(T_{m\times n})$. Also, the known results for this graph under $M \in \{cl, a, na\}$ are presented in Table 7.2.

Theorem 5.2.2. *The broadcast time of a torus network under the fully-adaptive model is as follows:*

$B_{fa}(T_{m\times n}) = \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor$, *if n and m are even,*

$B_{fa}(T_{m\times n}) = \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 1$, *if only one of m and n is even,*

$\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 1 \leq B_{fa}(T_{m\times n}) \leq \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 2$, *if m and n are odd.*

Proof. Observe that $B_{cl}(T_{m\times n}) = \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor$ if both n and m are even, and $\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 1$ otherwise (Farley & Hedetniemi, 1978). We will prove that these lower bounds are achievable under the fully-adaptive model unless both m and n are odd. To this aim, we will describe a simple broadcast scheme, σ, that achieves the above-mentioned bounds.

Similar to the broadcast scheme described for a grid, a vertex prioritizes its neighbors as follows: the above neighbor - the below neighbor - the right neighbor - and the left neighbor. Therefore, the ordering of a vertex (x, y) in σ is as follows: $l_{(x,y)} =< (x, (y+1) \mod n), (x, (y-1) \mod n), ((x+1) \mod m, y), ((x-1) \mod m, y) >$. Hereafter, we will show how to achieve the upper bound in each case.

1. When both n and m are even: Assume the originator is vertex (x, y). The message will be transmitted to its column in the first $\frac{n}{2}$ time units since one may look at each column as a cycle of length n. Afterward, during the next $\frac{m}{2}$ time units, all the vertices in column x will simultaneously inform their neighbor from right and left. Therefore, in this case, $B^{\sigma}_{fa}(T_{m\times n}) \leq \frac{n}{2} + \frac{m}{2}$.

2. When n is odd, and m is even, or vice versa: Without loss of generality, assume that n is odd and m is even. The analysis of the other case is very similar. Likewise to the previous case, if the originator is vertex (x, y), the vertices on column x will be informed in not more than $\frac{n+1}{2}$ time units. Note that in a cycle of odd length, the actual broadcast time is $\frac{n+1}{2}$. Afterward, an informed vertex in row i will take care of all the vertices in row i, and this process will be done in parallel for all informed vertices on row i, $1 \leq i \leq n$. This process will be finished in $\frac{m}{2}$ time units. Subsequently, in this case, $B^{\sigma}_{fa}(T_{m\times n}) \leq \frac{n+1}{2} + \frac{m}{2} = \lfloor\frac{n}{2}\rfloor + \lfloor\frac{m}{2}\rfloor + 1$.

3. When both n and m are odd: In this case, the same broadcast scheme, σ, will result in 1 step delay compared to the optimal time. The analysis is similar to the previous cases: In the first $\frac{n+1}{2}$ time units, the vertices on column x will be informed, and in the next $\frac{m+1}{2}$ time units the message will spread throughout the whole network. Therefore, $B^{\sigma}_{fa}(T_{m\times n}) \leq \frac{n+1}{2} + \frac{m+1}{2} = \lfloor\frac{n}{2}\rfloor + \lfloor\frac{m}{2}\rfloor + 2$. □

5.2.4 Cube Connected Cycles CCC_d

In this section, we study the fully-adaptive broadcast time of CCC_d. We follow the definitions and notations described earlier in section 2.5.11 for a CCC_d.

Theorem 5.2.3. $B_{fa}(CCC_d) = \lceil\frac{5d}{2}\rceil - 1$.

Proof. Since $B_{cl}(CCC_d) = \lceil\frac{5d}{2}\rceil - 1$, therefore, $B_{fa}(CCC_d) \geq \lceil\frac{5d}{2}\rceil - 1$. Hereafter, we will present a broadcast scheme σ for which $B^{\sigma}_{fa}(CCC_d) \leq \lceil\frac{5d}{2}\rceil - 1$.

The construction of the broadcast scheme σ is as follows: for each vertex, we will prioritize the lateral port against the backward and forward ports. In particular, consider a vertex $v \in V_{CCC_d}$

which is represented by the pair $\langle \alpha, i \rangle$, $\alpha \in \{0,1\}^d$ and $i \in \{0, 1, \cdots, d-1\}$. The ordering of this vertex in σ is as follows: $v :< \langle \alpha(i), i \rangle, \langle \alpha, i-1 \pmod{d} \rangle, \langle \alpha, i+1 \pmod{d} \rangle >$.

Now we will describe the process of achieving the upper bound on the broadcast time of a CCC_d under the fully-adaptive model using σ in two phases:

Phase 1: By time $t = 2d - 1$, there will be at least one informed vertex in each cycle. Soon in Theorem 7.1.2, we will prove that $B_{fa}(H_d) = d$. Thus, broadcasting within the embedded Hypercube will not take more than d time units. However, This process is delayed for $d-1$ time units in a CCC_d since each vertex has to use its backward port first to find an uninformed vertex v' whose lateral port is connected to an uninformed vertex. In other words, this phase is a similar process to broadcasting within H_d with the difference that each call to the other dimension in the H_d is replaced with two calls in CCC_d. Hence, during the first $2d-1$ time units, there will be at least one informed vertex in each cycle (the call at time $2d$ is unnecessary for this condition).

Phase 2: There are 2^d cycles of length d in a CCC_d. Note that broadcasting within a cycle of length d will not take more than $\lceil \frac{d}{2} \rceil$ time units. Therefore, using σ for an arbitrary CCC_d, the broadcasting will be finished for all cycles in parallel in the next $\lceil \frac{d}{2} \rceil$ time units. So, all vertices of the CCC_d will be informed in $2d - 1 + \lceil \frac{d}{2} \rceil = \lceil \frac{5d}{2} \rceil - 1$ time units. $\square$

It should be noted that, as it will be discussed shortly, there exists an infinitely large family of networks G, for which $B_{cl}(G) < B_{fa}(G)$. However, as seen in Theorem 5.2.3, this is not the case for CCC_d since we could achieve the same broadcast time under the fully adaptive model as that of classical. In particular, there is no additional call under the fully-adaptive model compared to the classical model for this family of interconnection networks using the suggested universal lists. To illustrate, the only situation in which a call is *"wasted"* under the fully-adaptive model compared to the classical model is when m informed vertices, denoted by $v_1, v_2, \cdots, v_m$, make m calls toward an uninformed vertex u at the same time t, simultaneously. We will shortly discuss why this situation will not happen using our broadcast scheme σ in a CCC_d under the fully-adaptive model.

Consider an uninformed vertex $u \in V_{CCC_d}$ which is represented by $< \alpha, i >$. This vertex has a degree of 3. We denote the neighbours of vertex u by v_1, v_2, and v_3. Without loss of generality, assume that vertices v_1 and v_2 are located on the same cycle as u, while v_3 is on another cycle, $\alpha(i)$.

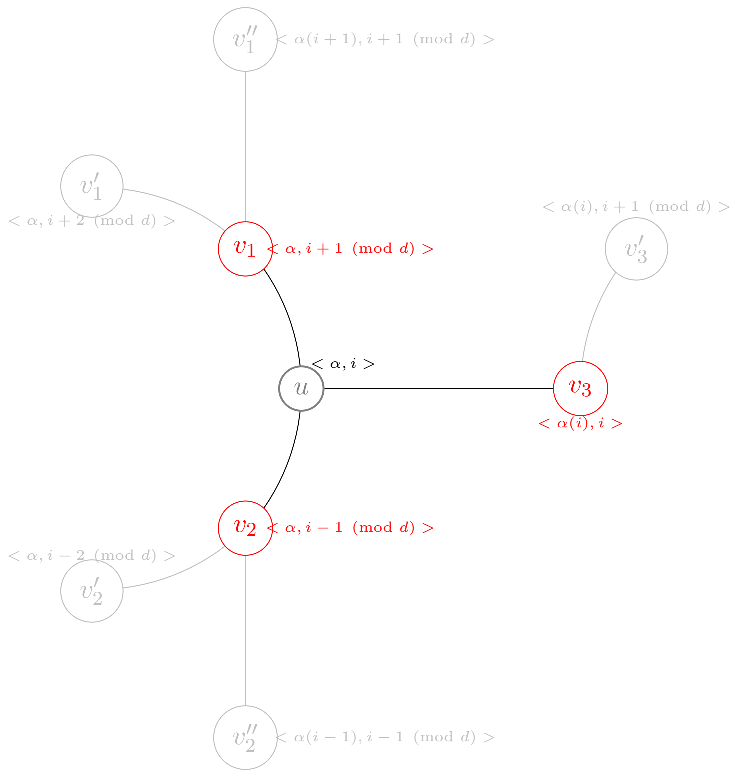

v_1''
$< \alpha(i+1), i+1 \pmod d >$
v_1'
$< \alpha, i+2 \pmod d >$
v_1
$< \alpha, i+1 \pmod d >$
$< \alpha(i), i+1 \pmod d >$
v_3'
$< \alpha, i >$
u
v_3
$< \alpha(i), i >$
v_2
$< \alpha, i-1 \pmod d >$
$< \alpha, i-2 \pmod d >$
v_2'
v_2''
$< \alpha(i-1), i-1 \pmod d >$

Figure 5.2: The relative location of an uninformed vertex u in an arbitrary CCC_d.

Therefore, we denote vertex v_1 by $< \alpha, i+1 \pmod d >$, vertex v_2 by $< \alpha, i-1 \pmod d >$, and vertex v_3 by $< \alpha(i), i >$. It should be mentioned that these three vertices are also connected to two other vertices, denoted by $v_1', v_1'', v_2', v_2'', v_3'$, and v_3''. The location of these vertices is portrayed in Figure 5.2.

Since in our broadcast scheme, σ, we prioritize the lateral port against the backward and forward ports, the broadcast scheme is as follows: $\sigma = \langle \cdots, v_1 :< v_1'', u, v_1' >, v_2 :< v_2'', v_2', u >, v_3 :< u, \cdots >, u :< v_3, v_2, v_1 >, \cdots \rangle$.

Also, note that since u is not informed, vertices v_1, v_2, and v_3 must have received the message from any of their neighbors at time $t-1$, but u. Denote the vertices that have sent the message to these vertices by $v_1^{(t-1)}$, $v_2^{(t-1)}$, and $v_3^{(t-1)}$. We are interested in locating these vertices in the CCC_d. Note that vertex v_3' must have made a call toward vertex v_3 at time $t-1$: $v_3^{(t-1)} = v_3'$. Thus, based on σ, vertex v_3 will send the message to vertex u at time t. Hereafter, we want to argue that neither v_1 nor v_2 will make a call toward u at time t.

In the ordering of vertex v_2, vertex u is the last node. So, v_2 will not send the message to vertex

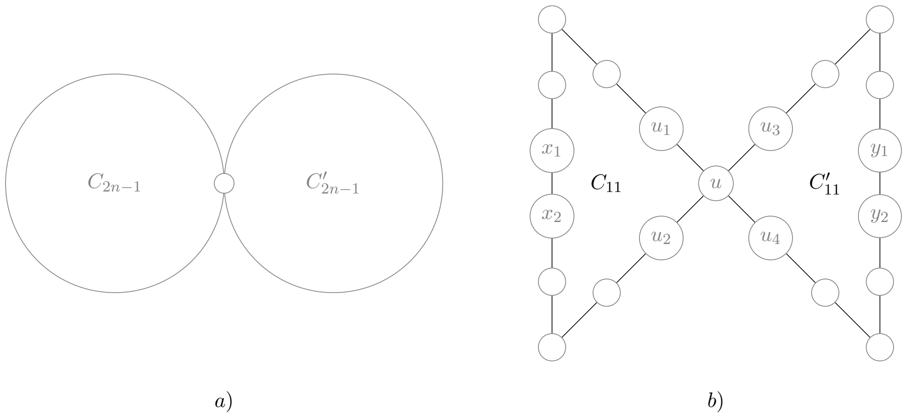


Figure 5.3: a) A graph G with $B_{cl}(G) < B_{fa}(G)$, b) An example with $n = 6$.

u, unless all of its neighbors are already informed, in which case that call is not considered to be *"wasted"* anymore (it will not delay the process). However, if vertex v_1 had received the message from vertex v''_1 at time $t-1$, it will skip this vertex on its list and make a call toward vertex u, while one of its neighbors, v'_1, might still be uninformed. We want to show that $v_1^{(t-1)} \neq v''_1$.

For the sake of contradiction, assume that vertex v''_1 had sent the message to vertex v_1 at time $t-1$. Note that our broadcast scheme, σ, is symmetric for all vertices of CCC_d since once a vertex gets informed, it will first use its lateral port and then its backward and forward ports. Therefore, if vertex v''_1 made a call toward vertex v_1 at time $t-1$, it implies that vertex v''_3 has sent the message to vertex v'_3 at time $t-1$. This contradicts with our assumption regarding vertex v_3 being informed at time $t-1$. Therefore, $v_1^{(t-1)} \neq v''_1$.

In conclusion, following our broadcast scheme σ for an arbitrary CCC_d under the fully-adaptive model, a call is not *"wasted"* compared to the classical model unless all neighbors of some vertices $v_1, \cdots, v_m$ are informed. In that case, they might all call the same vertex u simultaneously, but this will not delay the process as vertices v_i, $1 \leq i \leq m$, had no other choice but to remain idle.

5.2.5 Graphs with $B_{cl}(G) < B_{fa}(G)$

Based on the results presented so far, one might suspect that the broadcast times of a graph under classical and fully-adaptive models are always the same. In this section, we will provide an infinitely large family of graphs, G, for which $B_{cl}(G) < B_{fa}(G)$.

Proposition 5.2.1. *There exist arbitrarily large graphs G with $B_{cl}(G) < B_{fa}(G)$.*

Proof. The graph G is constructed by merging two cycles C_{2n-1} and C'_{2n-1} in one vertex (Figure 5.3-a). We will present some insights on the broadcast time of this graph using the example provided in Figure 5.3-b, in which the value of n is set to 6. Extending this result to the general case will be trivial. Denote the graph of Figure 5.3-b by G. Considering the classical model of broadcasting, observe that $B_{cl}(G) = B_{cl}(v \in \{x_1, x_2, y_1, y_2\}, G) = 11$. We will show that there is no universal list σ for which $B^{\sigma}_{fa}(G) = 11$.

We will prove this by contradiction. Suppose a scheme σ exists for which broadcasting under the fully-adaptive model finishes within 11 time units starting from any originator. It implies that $B^{\sigma}_{fa}(v \in \{x_1, x_2, y_1, y_2\}, G) \leq 11$. Now consider the list of vertex u under σ, or l_u. W.l.o.g suppose that l_u starts with u_1: $l_u =< u_1, \cdots >$. Take x_2 as the originator. Following any scheme, u will not be informed sooner than time 5. At time 6, it will pass the message to vertex u_1. Starting from time 7, it will start broadcasting within C'_{11} which will take at least $\lceil \frac{11}{2} \rceil = 6$ time units, so $B^{\sigma}_{fa}(x_2, G) > 11$, which contradicts the assumption. Therefore, there is no universal list σ for which $B^{\sigma}_{fa}(G) = 11$. More generally, For the graph given in Figure 5.3-a, $B_{cl}(G) = 2n - 1$, while $B_{fa}(G) > 2n - 1$. □

Chapter 6

Non-adaptive Broadcasting

This chapter will continue studying the problem of broadcasting with universal lists by focusing on the non-adaptive model in Section 6.1. We first provide the exact broadcast time of $k-$ary trees and binomial trees under the non-adaptive model. Then, we give an upper bound for the Complete Bipartite graph. Finally, we improve the general upper bound for trees that were initially presented in (Harutyunyan et al., 2011).

Later in Section 6.2, we study the relation between the universal lists model and messy broadcasting. This enables us to prove several general upper bounds for all models of universal lists for arbitrary graphs.

6.1 Results on the non-adaptive model

6.1.1 Complete k-ary trees $T_{k,h}$

Observe that for any tree T, $\lceil\frac{3\times diam(T)-1}{2}\rceil \leq B_{na}(T) \leq B_{cl}(T) + \lceil\frac{diam(T)}{2}\rceil$ (Harutyunyan et al., 2011). Thus, since $B_{cl}(T_{k,h}) = B_a(T_{k,h}) = kh + h - 1$ and $diam(T_{k,h}) = 2h$ (Hedetniemi et al., 1988), the following is established for an arbitrary $k-$ary tree:

Observation 6.1.1. *For any $k-$ary tree $T_{k,h}$, the broadcast time under the non-adaptive model is bounded as follows:*

$$\lceil\frac{6h-1}{2}\rceil \leq B_{na}(T_{k,h}) \leq kh + 2h - 1 \tag{20}$$

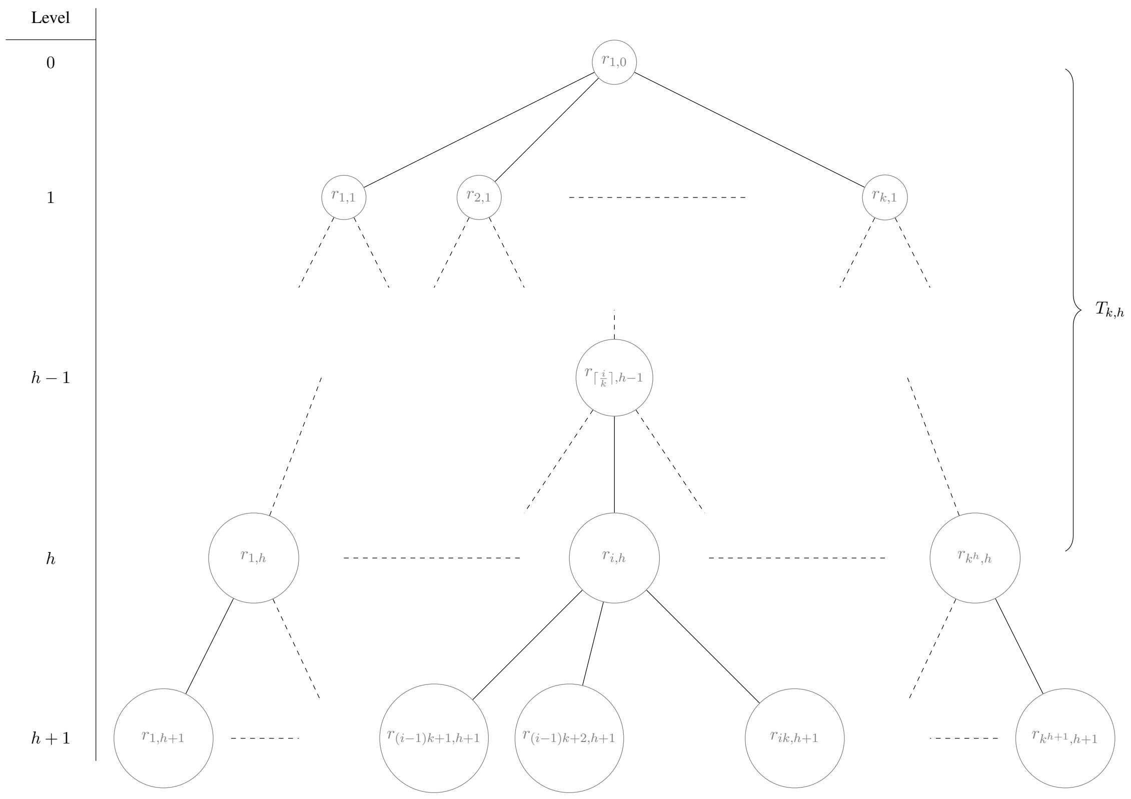


Figure 6.1: $T_{k,h+1}$, An arbitrary $k-$ary tree with height $h+1$. Observe that all nodes of this tree except for the leaves form a $k-$ary tree with height h.

Now, we will show that the upper bound of Equation (20) is tight. Denote a vertex of a $T_{k,h}$ by $r_{k',h'}$ in which k' is a number associated with the vertices at level h', from left to right: $1 \leq k' \leq k, 0 \leq h' \leq h$. Using this notation, the root of a $T_{k,h}$ tree is denoted by $r_{1,0}$. This vertex has k children, namely $r_{1,1}, r_{2,1}, \cdots, r_{k,1}$, from left to right. Also, the vertices at level h, or the leaves of $T_{k,h}$, are denoted by $r_{1,h}, r_{2,h}, \cdots, r_{k^h,h}$ (See Fig. 6.1).

Theorem 6.1.1. *For any complete $k-$ary tree, $B_{na}(T_{k,h}) = kh + 2h - 1$.*

Proof. Note that the broadcast scheme σ achieving the upper bound of Equation (20) could be realized by prioritizing the parent of each internal vertex against its children from left to right. Indeed, the list of a leaf node consists of its parent only, whereas the root sends the message to its children from left to right. According to Observation 6.1.1, it is essential to prove the lower bound: $B_{na}(T_{k,h}) \geq kh + 2h - 1$. We will do so by induction on h, the height of $T_{k,h}$.

For the base case, when $h = 1$, consider an arbitrary $T_{k,1}$, in which the root $r_{1,0}$ has k children

$r_{1,1}, r_{2,1}, \cdots, r_{k,1}$, which is a star with the center being the root vertex $r_{1,0}$. It follows from (Diks & Pelc, 1996) that: $B_{na}(T_{k,1}) \geq k+1 = k \cdot 1 + 2 \cdot 1 - 1$. For the induction hypothesis, assume $B_{na}(T_{k,h}) \geq kh + 2h - 1$ for an arbitrary h. Now, it must be proved that $B_{na}(T_{k,h+1}) \geq k(h+1) + 2(h+1) - 1 = kh + 2h + k + 1$. To this aim, we will prove under any scheme σ, $\exists v$: $B_{na}(v, T_{k,h+1}) \geq kh + 2h + k + 1$. Based on the ordering of the vertices at level h under σ, there are two possible cases:

(1) All the vertices at level h first inform their parent, then their children in some order:

(2) There is at least a vertex (such as $r_{i,h}$) which informs some of its children, then its parent, and afterward, the rest of its children, if any.

Consider vertex $r_{1,h+1}$ as the originator. At time $t = 1$ this vertex informs vertex $r_{1,h}$ under any scheme σ as that is the only choice. Since $B_{na}(T_{k,h}) \geq kh + 2h - 1$, there exists a vertex at level h which receives the message at the last time unit, or at $t = kh + 2h$. Denote this vertex by $r_{i,h}$. Observe that until time unit $t = kh + 2h$, at least all vertices of the first h level in $T_{k,h+1}$ are informed in addition to the originator $r_{1,h+1}$. However, the children of vertex $r_{i,h}$, or $r_{j,h+1}, \forall j$: $(i-1)k + 1 \leq j \leq ik$ cannot be informed since their parent was informed at the last time unit. Depending on the ordering of vertex $r_{i,h}$ under the arbitrary scheme σ, two cases may arise:

(1) If vertex $r_{i,h}$ first sends the message back to its parent, and then to all of its children in some order: In this case, at time $t = kh + 2h + 1$, vertex $r_{i,h}$ re-sends the message to vertex $r_{\lceil \frac{i}{k} \rceil, h-1}$. Then, at the next k time units, $r_{i,h}$ will send the message to its children. By the time $t = kh + 2h + k + 1$, the last child will be informed. It implies that: $B_{na}(r_{1,h+1}, T_{k,h+1}) \geq kh + 2h + k + 1 (*)$.

(2) If vertex $r_{i,h}$ sends the message to some of its children, then its parent, and lastly to the rest of its children in some order: in this case, denote the first child that received the message from vertex $r_{i,h}$ by vertex $r_{i_1,h+1}$. Now, we will consider $r_{i_1,h+1}$ to be the originator.

At time $t = 1$, $r_{i_1,h+1}$ informs vertex $r_{i,h}$ since this is its only choice. Then at time $t = 2$, vertex $r_{i,h}$ sends the message back to vertex $r_{i_1,h+1}$, according to its broadcast scheme. at time $t \geq 3$, $r_{i,h}$ may send the message to its parent and start the broadcast process within the subtree $T_{k,h}$. Therefore, there exists a vertex, say $r_{j,h}$, that will be informed at time $t \geq 3 + kh + 2h - 1 = kh + 2h + 2$. This vertex has k children that must be informed under any broadcast scheme, including σ. This process will be finished at time $t \geq kh + 2h + k + 2$. Subsequently, $B_{na}(r_{i_1,h+1}, T_{k,h+1}) \geq kh + 2h + k + 2(**)$.

From $(*)$ and $(**)$, $B_{na}(T_{k,h+1}) \geq kh + 2h + k + 1$. Therefore, for any $k-$ary tree, $B_{na}(T_{k,h}) = kh + 2h - 1$. □

6.1.2 Binomial Tree T_d

The broadcast time of a binomial tree under the non-adaptive model could be proved using the following proposition:

Proposition 6.1.1. *For any binomial tree, $B_{na}(T_d) = 3d - 2$.*

Proof. Note that $B_{cl}(T_d) = diam(T_d) = 2d - 1$. By applying the general lower bound for the trees we get $B_{na}(T_d) \geq \lceil \frac{3\times(2d-1)-1}{2} \rceil = 3d - 2$. Furthermore, applying the first upper bound proposed in (Diks & Pelc, 1996), observe that: $B_{na}(T_d) \leq B_{cl}(T_d) + \lfloor \frac{B_{cl}(T_d)}{2} \rfloor = 3d - 2$, whether d is even or odd. □

6.1.3 Complete Bipartite graph $K_{m\times n}$

Although there are several real-world applications for this family of networks, from graph search in quantum computing (Rhodes & Wong, 2019) to enhancing task distribution (M.-s. Li, Xue, & Liu, 2019), to the best of our knowledge, there is a lack of a comprehensive study for this network considering the broadcasting problem, even under the classical model. Subsequently, in this part, we begin by proving the broadcast time of $K_{m\times n}$ under the classical model. Afterward, we will present a non-trivial upper bound for $K_{m\times n}$ under the non-adaptive model.

Theorem 6.1.2. *For any complete bipartite graph $K_{m\times n}$,*

$$B_{cl}(K_{m\times n}) = \lceil \log n \rceil + 1 + \max\{\lceil \frac{m - 2^{\lceil \log n \rceil}}{n} \rceil, 0\} \tag{21}$$

Proof. We begin by proving the **lower bound**. Observe that Equation (21) could be simplified to $B_{cl}(K_{m\times n}) = \lceil \log n \rceil + 1$ when $n = m$. For this particular case, note that the classical broadcast time of any graph with $|V|$ vertices is lower bounded by $\lceil \log |V| \rceil$. Since when $m = n$ for a complete bipartite graph, $|V| = 2n$, the lower bound is true.

For the case when $m > n$, regardless of the broadcast scheme, at time $t = 1$, there cannot be more than one informed vertex in both partitions. At each time unit, an informed vertex in a partition may inform a new vertex in the other partition. So, the number of informed vertices doubles each time. Therefore, at least $\lceil \log n \rceil + 1$ time units are required to inform n vertices in both partitions. In other words, during the first $\lceil \log n \rceil + 1$ time units, we will have at most $2^{\lceil \log n \rceil}$ informed vertices in partition M, while all n vertices in partition N are informed. Hence, there could be at most $m - 2^{\lceil \log n \rceil}$ uninformed vertices in partition M. In each time unit, at most n vertices will make a call to inform one of these uninformed vertices, so at least $\lceil \frac{m-2^{\lceil \log n \rceil}}{n} \rceil$ time units are necessary to inform the vertices in partition M completely. The lower bound for the case $m > n$ follows.

For the **upper bound**, we will provide an algorithm A that achieves this bound. For the case $m = n$, algorithm A begins with informing one vertex in the other partition at the first time unit. Then, during the next $\lceil \log n \rceil$ time units, each informed vertex will make a call to inform a new vertex in the other partition. Therefore, all vertices in both partitions are informed in $\lceil \log n \rceil + 1$ time units.

For the case when $m > n$, at time $t = 1$, a call will be made from the partition containing the originator to the other partition. Also, during the next $\lceil \log n \rceil$ time units, we will make a call from an informed vertex to another uninformed vertex in the other partition. Note that if $n \neq 2^i$, then some of the vertices in partition M will remain idle at time $\lceil \log n \rceil + 1$. At time $t = \lceil \log n \rceil + 1$ all vertices of partition N are informed, and in the other partition there will be no more than $m - 2^{\lceil \log n \rceil}$ uninformed vertices. Starting from time unit $t = \lceil \log n \rceil + 2$ until the last time unit, informed vertices of partition M will be idle, while all vertices from partition N will make a call to inform one vertex in the other partition. Since there were $m - 2^{\lceil \log n \rceil}$ uninformed vertices, this process will not take more than $\lceil \frac{m-2^{\lceil \log n \rceil}}{n} \rceil$ time steps. The upper bound follows. $\square$

Hereafter, an upper bound for the broadcast time of a complete bipartite graph under the non-adaptive model is suggested. Our upper bound proof follows the idea of the broadcast scheme presented in (Diks & Pelc, 1996) for Complete graphs.

Theorem 6.1.3. *For any complete bipartite graph $K_{m\times n}$,*

$$B_{na}(K_{m\times n}) \leq \lceil\log n\rceil+1+\max\{\lceil\frac{m-2^{\lceil\log n\rceil}}{n}\rceil,0\}+3\times\lceil\sqrt{\lceil\log n\rceil+1+\max\{\lceil\frac{m-2^{\lceil\log n\rceil}}{n}\rceil,0\}}\rceil \tag{22}$$

Proof. Let $t_1 = \lceil\log n\rceil + 1 + \max\{\lceil\frac{m-2^{\lceil\log n\rceil}}{n}\rceil, 0\}$. We will present a broadcast scheme σ in which the non-adaptive broadcasting finishes within $t_1 + 3\lceil\sqrt{t_1}\rceil$ time units.

Suppose $m \geq n$. Fix a node v in the graph. Take an optimal broadcast scheme originating from node v under the classical model, and denote it by σ'. The ordering of an arbitrary vertex u in σ' is denoted by l'_u. Further suppose vertex v sends the message to vertex r at the first time unit. Observe that the partition of vertices v and r will not be the same. Now we will construct a broadcast scheme σ for the non-adaptive model using σ':

(1) for vertex v: $l_v = l'_v$.

(2) for each vertex $u \neq v$, if u is not in the same partition as v:

(a) if $|l'_u| \geq \lceil\sqrt{t_1}\rceil$: l_u is the same as l'_u, but vertex v is added at position $\lceil\sqrt{t_1}\rceil$.

(b) if $|l'_u| < \lceil\sqrt{t_1}\rceil$: l_u is the same with l'_u, and vertex v is added at the end of the ordering.

(3) for each vertex $u \neq v$, if u is in the same partition as v:

(a) if $|l'_u| \geq \lceil\sqrt{t_1}\rceil$: l_u is the same as l'_u, but vertex r is added at position $\lceil\sqrt{t_1}\rceil$.

(b) if $|l'_u| < \lceil\sqrt{t_1}\rceil$: l_u is the same with l'_u, and vertex r is added at the end of the ordering.

First we will prove that $B^{\sigma}_{na}(v, K_{m\times n}) \leq t_1 + \lceil\sqrt{t_1}\rceil$. Denote the broadcast tree obtained from the classical model originating from v by T. Consider any path p on T and denote the vertices on this path by $P = v_0, v_1, \cdots, v_k$. Clearly, $v_0 = v$. Denote by t_i the position of vertex v_{i+1} on the list of vertex v_i under the classical model, or l'_{v_i}. Note that σ' is an optimal scheme. Therefore, the sum of the position of a vertex on the previous vertex's list cannot surpass the optimal broadcast time because the last call from v_{k-1} to v_k must be made not later than time t_1. Thus, $\sum_{i=0}^{k-1} t_i \leq t_1$.

Now we discuss the same process but using σ as the broadcast scheme under the non-adaptive model. The calls that were made on path P will not be delayed more than one time unit for each of them since we only added one vertex (either v or r) for constructing scheme σ. If a call from v_i to v_{i+1} is delayed for one time unit, we will set a variable δ_i to 1, and otherwise to 0. Observe that all vertices of the $K_{m\times n}$ will be informed when the originator is vertex v in at most $\sum_{i=0}^{k-1}(t_i + \delta_i)$ time units.

For calculating δ_i, note that there could not be more than $\lceil\sqrt{t_1}\rceil$ calls that are delayed in a path P by one time unit. In the worst case, all calls are delayed by one time unit. It means that in all the calls placed in P, v_i must inform v_{i+1} not sooner than time $\lceil\sqrt{t_1}\rceil$ (otherwise, the call will not be delayed). Even in this worst case, the number of calls on P cannot exceed $\lceil\sqrt{t_1}\rceil$ (since the whole process is finished in t_1 time units under the classical model). Hence, $\sum_{i=0}^{k-1} \delta_i \leq \lceil\sqrt{t_1}\rceil$. It is concluded that $\sum_{i=0}^{k-1}(t_i + \delta_i) \leq t_1 + \lceil\sqrt{t_1}\rceil$. Thus, $B_{na}^{\sigma}(v, K_{m\times n}) \leq t_1 + \lceil\sqrt{t_1}\rceil$. $(*)$

Lastly, two types of originators must be considered:

- If the originator is a vertex $u \neq v$, and u is not in the same partition as v: note that based on the position of vertex v on l_u, vertex v will be informed by some vertex at time $\lceil\sqrt{t_1}\rceil$, the latest. From that moment, we will follow the same procedure as mentioned above. Therefore, $\forall u \neq v,$ if the partition of u and v are different : $B_{na}^{\sigma}(u, K_{m\times n}) \leq t_1 + 2 \times \lceil\sqrt{t_1}\rceil$. $(**)$

- If the originator is a vertex $u \neq v$, and u is in the same partition as v: note that based on the position of vertex r on l_u, vertex r will be informed by some vertex at time $\lceil\sqrt{t_1}\rceil$, the latest. Also, using l_r, vertex v will be informed at most $\lceil\sqrt{t_1}\rceil$ time units later. From that moment, we will follow the same procedure as mentioned above. Therefore, $\forall u \neq v,$ if the partition of u and v are the same : $B_{na}^{\sigma}(u, K_{m\times n}) \leq t_1 + 3 \times \lceil\sqrt{t_1}\rceil$. $(***)$

From $(*)$, $(**)$, and $(***)$, Equation (22) is valid for the case when $m \geq n$. □

It is worth mentioning that $B_{fa}(K_{m\times n})$ and $B_a(K_{m\times n})$ are also bounded by the same values as Equation (22), due to Theorem 5.1.1.

6.1.4 A general Upper bound for Trees

An upper bound on the broadcast time for an arbitrary tree under the non-adaptive model is presented in (Harutyunyan et al., 2011) as follows: $B_{na}(T) \leq B_{cl}(T) + \lceil \frac{diam(T)}{2} \rceil$, where $diam(T)$ is the diameter of the tree T. We will improve this upper bound by 1 when the diameter is an odd number. Our upper bound also improves the upper bound presented in (Diks & Pelc, 1996).

Theorem 6.1.4. $B_{na}(T) \leq B_{cl}(T) + \lfloor \frac{diam(T)}{2} \rfloor$.

Proof. The proof is similar to the proof in (Harutyunyan et al., 2011). When $diam(T)$ is an even number, the statement is trivially concluded from Theorem 3.2. of (Harutyunyan et al., 2011). In what follows, the diameter is considered to be an odd number.

The broadcast center of a graph is the set of vertices with the minimum broadcast time. Denote one of the broadcast centers of T by u. Further suppose T is rooted at vertex u. Let σ' be an optimal broadcast scheme originating from u in T under the classical model. Also, l'_v denotes the ordering of vertex v under broadcast scheme σ'. We will construct another broadcast scheme σ for the non-adaptive model with: $B^{\sigma}_{na}(T) \leq B_{cl}(T) + \lfloor \frac{diam(T)}{2} \rfloor$. Let $l_u = l'_u$. For a vertex v other than u, we will construct l_v by prepending the parent of v to the list l'_v. In other words, once a vertex v gets informed, it will call its parent first and then proceeds to its children in an ordering identical to σ'.

Now fix an arbitrary originator v. There exists (at least) one vertex that receives the message at the last time unit under the broadcast scheme σ originating from v. Denote this vertex by x. Let $b^{\sigma}_{na}(v, x)$ denote the time steps required for a message to reach x once v is informed under the broadcast scheme σ. By definition of vertex x, $B^{\sigma}_{na}(v, T) = b^{\sigma}_{na}(v, x)$.

Let y be the first common ancestor of v and x in the tree T rooted at u. Also, let T_y be the subtree rooted at y. Further, suppose the distance of two vertices v_1 and v_2 is denoted by $d(v_1, v_2)$. Observe that:

$$B^{\sigma}_{na}(v, T) = b^{\sigma}_{na}(v, x) = d(v, y) + 1 + B_{cl}(y, T_y) + d(y, x) - 1 \qquad (23)$$

It takes $d(v, y)$ time steps for the message to reach to y from originator v. Afterward, vertex y will send the message to its parent following our broadcast scheme σ. Then y will follow the broadcast scheme identical to the classical model and inform its subtree within the next $B_{cl}(y, T_y)$ time units.

However, the vertices on the path connecting y to x will delay this process by sending to their parent first. This delays the process $d(y, x) - 1$ time units since x, being the last vertex that gets informed, will not send the message back to its parent.

Note that $d(v, y) + d(y, x) \leq diam(T)$ for any three vertices v, y, and x, when y is on the path connecting v to x. Moreover, observe that $\min\{d(v, y), d(y, x)\} \leq \lfloor \frac{diam(T)}{2} \rfloor$ when diameter is an odd number. Without loss of generality, assume that $d(v, y) \leq d(y, x)$. Therefore, $d(v, y) \leq \lfloor \frac{diam(T)}{2} \rfloor$:

$$B^{\sigma}_{na}(v, T) \leq \lfloor \frac{diam(T)}{2} \rfloor + B_{cl}(y, T_y) + d(y, x) \tag{24}$$

Note that for any vertex y and the broadcast center u: $B_{cl}(y, T_y) \leq B_{cl}(u, T) = b_{cl}(T)$, where $b_{cl}(T)$ is the minimum broadcast time of any vertex in tree T. Also, since the unique path connecting u to x must travel through y: $d(y, x) \leq d(u, x)$. Thus,

$$\begin{aligned} B^{\sigma}_{na}(v, T) &\leq \lfloor \frac{diam(T)}{2} \rfloor + b_{cl}(T) + d(u, x) \\ &= \lfloor \frac{diam(T)}{2} \rfloor + B_{cl}(x, T) \end{aligned} \tag{25}$$

The last step follows from Slater et. al (Slater et al., 1981): "for a central vertex u and an arbitrary vertex x in a tree T: $B_{cl}(x, T) = d(x, u) + b_{cl}(T)$". Lastly, since $B_{cl}(x, T) \leq B_{cl}(T)$ for any vertex x, and the originator v was chosen randomly:

$$B_{na}(T) \leq \lfloor \frac{diam(T)}{2} \rfloor + B_{cl}(T) \tag{26}$$

□

By this analysis, we managed to save one time unit, hence, achieving a tight upper bound: consider a binomial tree T_d, where $B_{cl}(T_d) = diam(T_d) = 2d - 1$. According to Theorem 6.1.4, $B_{na}(T_d) \leq \lfloor \frac{2d-1}{1} \rfloor + 2d - 1 = 3d - 2$. However, by Proposition 6.1.1, we know that this is the optimal value.

Not only does this result improve Theorem 3.2. of (Harutyunyan et al., 2011), but it will also achieve a tighter bound compared to Theorem 2.1. of (Diks & Pelc, 1996), in which the authors proved that $B_{na}(T) \leq \lfloor \frac{B_{cl}(T)}{2} \rfloor + B_{cl}(T)$[1]. The following theorem summarizes the best bounds for the broadcast time of a tree under the non-adaptive model:

Theorem 6.1.5. *For a Tree T:*

$$\max\left\{B_{cl}(T)+1, \lceil \frac{3.diam(T)-1}{2} \rceil\right\} \leq B_{na}(T) \leq \min\left\{B_{cl}(T)+\lfloor \frac{diam(T)}{2} \rfloor, b_{cl}(T)+diam(T)\right\} \tag{27}$$

Each of these bounds is tight for some trees. For instance:

- For Star S_n: $B_{na}(S_n) = B_{cl}(S_n) + 1$,
- For Path P_n: $B_{na}(P_n) = \lceil \frac{3 \cdot diam(P_n)-1}{2} \rceil$,
- For Binomial tree T_d: $B_{na}(T_d) = B_{cl}(T_d) + \lfloor \frac{diam(T_d)}{2} \rfloor$,
- For Fork graph[2] $F_{n,\frac{n}{2}-1}$: $B_{na}(F_{n,\frac{n}{2}-1}) = b_{cl}(F_{n,\frac{n}{2}-1}) + diam(F_{n,\frac{n}{2}-1})$.

6.2 Comparison of universal list and messy broadcasting

We start with a primary result:

Lemma 6.2.1. *For any graph G:*

$B_{fa}(G) \leq t_1(G)$.

$B_a(G) \leq t_2(G)$.

$B_{na}(G) \leq t_3(G)$.

Proof. We begin by proving the first statement. For graph G, consider an arbitrary broadcast scheme σ under the fully-adaptive model. Fix the originator u, and then start broadcasting from u using σ. Once performing the broadcasting, if a vertex v skips v' on its list at time t, it means that vertex v' was informed by time unit $t-1$. Now we may use the same scheme σ for broadcasting under

[1]Note that $diam(T) \leq B_{cl}(T)$ for any tree T.

[2]A Fork graph $F_{n,k}$ consists of n nodes: A Path of $n-k$ vertices and k pendant vertices that are attached at one end point of the path.

model M_1. Note that any vertex such as v will skip its neighbor v' at time t since v' must have been informed at any time $t' < t$ following σ.

In other words, assume that the broadcast time of graph G under the fully-adaptive model using σ is denoted by $B^{\sigma}_{fa}(G)$. Note that the value of $B^{\sigma}_{fa}(G)$ is a lower bound on $t_1(G)$, since $t_1(G)$ is the maximum value among all possible broadcast schemes. Thus, $B^{\sigma}_{fa}(G) \leq t_1(G)$. Since $B_{fa}(G) \leq B^{\sigma}_{fa}(G)$, the result follows.

The proofs of the second and the third statements go in the same direction. To illustrate, the value realized by following an arbitrary broadcast scheme σ under the adaptive model is always a valid candidate for model M_2: $B_a(G) \leq B^{\sigma}_{a}(G) \leq t_2(G)$. Observe that the nature of the vertices that are to be skipped under the adaptive model is the same as that of model M_2. The same is also true for the non-adaptive model and messy model M_3: $B_{na}(G) \leq B^{\sigma}_{na}(G) \leq t_3(G)$. □

We will also give some graphs for which the inequalities of Lemma 6.2.1 are on their boundaries. For the equality case, consider path P_n on n vertices. It is proved that $B_{fa}(P_n) = B_a(P_n) = n-1$. Also, from (Harutyunyan & Liestman, 1998) we know that $t_1(P_n) = t_2(P_n) = n-1$. Also, consider Star S_n for non-adaptive model where $B_{na}(S_n) = n$ (Diks & Pelc, 1996). The same value is achieved under model $M_3 : t_3(S_n) = n$. This shows that the upper bounds of Lemma 6.2.1 cannot be improved in general. Besides, one may suspect that the values of Lemma 6.2.1 are always equal. As a counter-example, consider complete graph K_n on n vertices. From (Harutyunyan & Liestman, 1998) one may notice that $t_1(K_n) = t_2(K_n) = t_3(K_n) = n-1$. However, a non-trivial upper bound presented in (Diks & Pelc, 1996) suggests that $B_{fa}(K_n) \leq B_a(K_n) \leq B_{na}(K_n) \leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil$.

Additionally, we argue that there is no relation between model M_2 and the non-adaptive model or between model M_1 and the adaptive model. As an example regarding the first case, consider Path P_n and the Complete graph K_n for which: $B_{na}(K_n) \leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil < n-1 = t_2(K_n)$ and $B_{na}(P_n) = \lceil \frac{3n}{2} \rceil - 2 > n-1 = t_2(P_n)$ (Diks & Pelc, 1996). We also conjecture that the same is true for the second case, though we are not able to find a simple graph G with $B_a(G) > t_1(G)$.

Using Lemma 6.2.1, several upper bounds on $B_M(G)$ for $M \in \{fa, a, na\}$ could be achieved for any connected graph G with n vertices and m edges, diameter $diam(G)$, and the maximum degree of Δ:

Corollary 6.2.1. *$B_{na}(G) \leq 2m - 1$, and $B_{fa}(G) \leq B_a(G) \leq m$.*

Corollary 6.2.2. *$B_{na}(G) \leq diam(G) \cdot \Delta$, and $B_{fa}(G) \leq B_a(G) \leq diam(G) \cdot (\Delta - 1) + 1$.*

The truth of Corollary 6.2.1 can be realized by noticing that an edge could be used at most twice under the non-adaptive model, once in each direction. However, once the last vertex is informed, the process will stop, and it will not call back using the same edge. For the fully-adaptive and adaptive model, on the other hand, an edge (u, v) may be utilized at most once, either for sending from u to v or vice versa. The result follows. Besides, the truth of Corollary 6.2.2 could be realized by utilizing Corollary 2.1 of (Harutyunyan & Liestman, 1998) which proves the same inequality on $t_i(G)$ for $i = 1, 2, 3$.

Chapter 7

Broadcast Graphs under the Fully-adaptive Model

In this chapter, we propose several graphs for the universal list model for which broadcasting could be finished as quickly as theoretically possible from any originator in Section 7.1. Additionally, we prove that a hypercube H_d is a graph with minimum possible edges for any value of 2^k to achieve the minimum broadcast time. Also, in Section,7.2, we summarize the results presented in the last three chapters.

7.1 Broadcast graphs under the fully-adaptive model

A *broadcast graph (bg)* is a graph $G = (V, E)$ in which broadcasting could be completed within the minimum possible time starting from any originator. In the classical model, the minimum possible time is $\lceil \log n \rceil$ for a graph on n vertices; thus, for an arbitrary broadcast graph G: $\forall u \in V(G) : B_{cl}(u, G) = \lceil \log n \rceil$. Additionally, a *minimum broadcast graph (mbg)* is the *bg* with the minimum possible number of edges. We denote by $B^{(M)}(n)$ the broadcast function for an arbitrary value of n under model $M \in \{cl, fa\}$, which is the number of edges associated with the mbg for n. Finding mbg's for different values of n is quite vital since they represent the networks with minimum cost in which broadcasting could be performed as quickly as possible from any originator.

Although this problem is well studied under $M = cl$ for several values of n, there is no comparative study under universal lists models. Despite the considerable effort, an *mbg* is known only for very few n under the classical model. In particular, Hypercubes H_k (Farley et al., 1979) and Knödel graphs $W_{k,2^k}$ (Hedetniemi et al., 1988; Knödel, 1975) are mbg's for $n = 2^k$, while the latter family is also an mbg for $n = 2^k - 2$ (Dinneen et al., 1991; Khachatrian & Harutounian, 1990). The value of $B^{(cl)}(n)$ is also known for small values of $n \leq 32$ except for $n = 23, 24, 25$ (Barsky, Grigoryan, & Harutyunyan, 2014; Bermond et al., 1992; Farley et al., 1979; Maheo & Saclé, 1994; Saclé, 1996).

7.1.1 *mbg*'s for some values of $n \leq 16$

In what follows, the mbg's under the fully-adaptive model for all values of $n \leq 10$ are presented. Moreover, we provide upper bounds on the value of $B^{(fa)}(n)$ for $11 \leq n \leq 14$. We first prove a lower bound on the value of $B^{(fa)}(n)$:

Lemma 7.1.1. *If there is a graph G on n vertices for which $B_{fa}(G) = \lceil \log n \rceil$, then $B^{(cl)}(n) \leq B^{(fa)}(n)$.*

Proof. Consider an arbitrary graph G_1 with n_1 vertices and m_1 edges under fully-adaptive model such that $B_{fa}(G_1) = \lceil \log n_1 \rceil$. Then, G_1 is a broadcast graph. Observe that for any graph G with n vertices, $\lceil \log n \rceil \leq B_{cl}(G) \leq B_{fa}(G)$. Therefore, $B_{cl}(G_1) = \lceil \log n_1 \rceil$, and G_1 is a broadcast graph under the classical model as well. Thus, m_1 is an upper bound for $B^{(cl)}(n_1)$. The result follows. $\square$

The broadcast time of a Cycle C_n under the fully-adaptive model is $\lceil \frac{n}{2} \rceil$. This value is as low as $\lceil \log n \rceil$ for $n \leq 6$. Consequently, since C_n is also an mbg under the classical model and due to Lemma 7.1.1, the following corollary is concluded:

Corollary 7.1.1. *Cycle C_n is mbg under the fully-adaptive model for $4 \leq n \leq 6$.*

Hereafter, we will prove that the mbg's for classical broadcasting, presented in (Farley et al., 1979), are also mbg's for the fully-adaptive model for $n = 7, 9$, and 10. Figure 7.1,a-c illustrate the mbg's for these values under the fully-adaptive model, which are identical to that of the classical

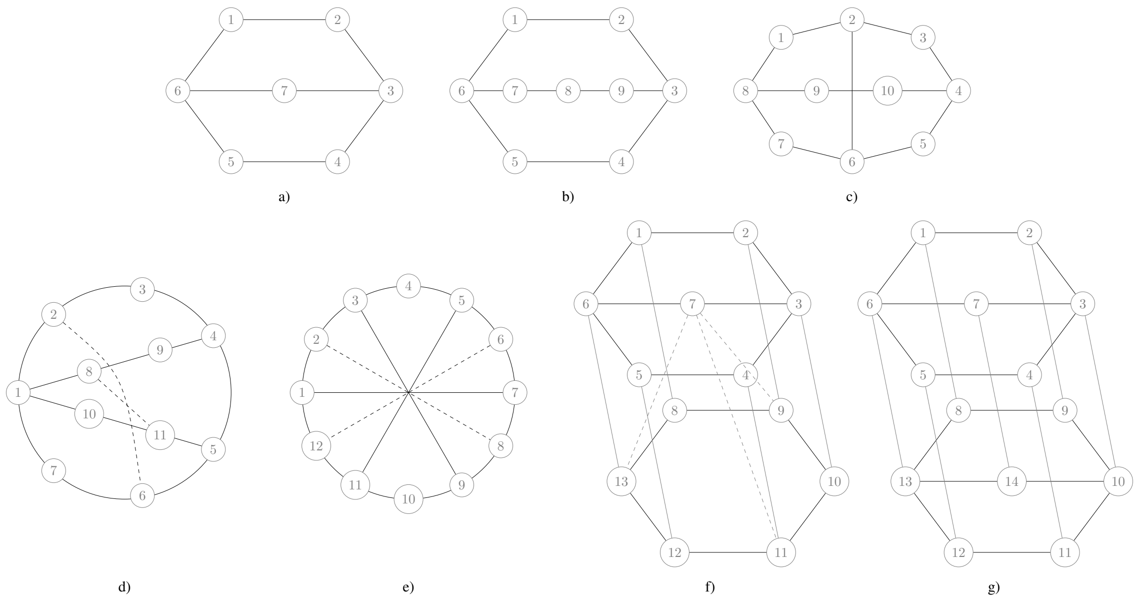


Figure 7.1: a-c)*mbg*'s on 7,9, and 10 vertices under the fully-adaptive model. d-g) *bg*'s on 11,12,13 and 14 vertices under the fully-adaptive model.

model. The universal lists achieving a broadcast time of $\lceil \log n \rceil$ under the fully-adaptive model for the mbg's on n vertices presented in Figure 7.1 are as follows:

- $\sigma_7 = \{1 :< 6, 2 >, 2 :< 3, 1 >, 3 :< 4, 2, 7 >, 4 :< 3, 5 >, 5 :< 6, 4 >, 6 :< 1, 5, 7 >, 7 :< 3, 6 >\}$,
- $\sigma_9 = \{1 :< 6, 2 >, 2 :< 3, 1 >, 3 :< 4, 2, 9 >, 4 :< 3, 5 >, 5 :< 6, 4 >, 6 :< 1, 5, 7 >, 7 :< 6, 8 >, 8 :< 7, 9 >, 9 :< 3, 8 >\}$,
- $\sigma_{10} = \{1 :< 2, 8 >, 2 :< 1, 3, 6 >, 3 :< 2, 4 >, 4 :< 5, 10, 3 >, 5 :< 4, 6 >, 6 :< 2, 5, 7 >, 7 :< 8, 6 >, 8 :< 9, 1, 7 >, 9 :< 8, 10 >, 10 :< 9, 4 >\}$.

This problem becomes more difficult for $n \geq 11$ under the fully-adaptive model. For instance, our exhaustive search on more than 4000 broadcast schemes on mbg's with $n = 11, \cdots, 15$ vertices did not result in a broadcast scheme that achieves $B^{\sigma}_{fa}(G) = 4$. However, by adding a few more edges to the mbg's on the classical model, an upper bound on the value of $B^{(fa)}(n)$ is obtained for $11 \leq n \leq 14$. These bg's are presented in Figure 7.1,d-g, respectively, while the broadcast schemes achieving time 4 are presented in what follows. Table 7.1 summarizes the results of this section.

Table 7.1: The known values of $B^{(fa)}(n)$ for $n < 15$

n	3	4	5	6	7	8	9	10	11	12	13	14
Lower bound on $B^{(fa)}(n)$	2	4	5	6	8	12	10	12	13	15	18	21
Upper bound on $B^{(fa)}(n)$	2	4	5	6	8	12	10	12	15	17	23	23

- $\sigma_{11} = \{1 :< 7, 8, 2, 10 >, 2 :< 6, 3, 1 >, 3 :< 4, 2 >, 4 :< 3, 5, 9 >, 5 :< 11, 4, 6 >, 6 :< 5, 2, 7 >, 7 :< 6, 1 >, 8 :< 9, 11, 1 >, 9 :< 8, 4 >, 10 :< 11, 1 >, 11 :< 5, 10, 8 >\}$,

- $\sigma_{12} = \{1 :< 2, 12, 7 >, 2 :< 8, 3, 1 >, 3 :< 2, 9, 4 >, 4 :< 3, 5 >, 5 :< 11, 6, 4 >, 6 :< 12, 7, 5 >, 7 :< 6, 1, 8 >, 8 :< 2, 9, 7 >, 9 :< 8, 3, 10 >, 10 :< 11, 9 >, 11 :< 12, 5, 10 >, 12 :< 6, 1, 11 >\}$,

- $\sigma_{13} = \{1 :< 2, 6, 8 >, 2 :< 1, 3, 9 >, 3 :< 2, 4, 7, 10 >, 4 :< 3, 5, 11 >, 5 :< 4, 6, 12 >, 6 :< 5, 1, 7, 13 >, 7 :< 9, 6, 11, 3, 13 >, 8 :< 13, 9, 1 >, 9 :< 8, 10, 2, 7 >, 10 :< 9, 11, 3 >, 11 :< 10, 12, 4, 7 >, 12 :< 11, 13, 5 >, 13 :< 12, 8, 6, 7 >\}$,

- $\sigma_{14} = \{1 :< 6, 2, 8 >, 2 :< 3, 1, 9 >, 3 :< 4, 2, 7, 10 >, 4 :< 3, 5, 11 >, 5 :< 6, 4, 12 >, 6 :< 1, 5, 7, 13 >, 7 :< 3, 6, 14 >, 8 :< 13, 9, 1 >, 9 :< 10, 8, 2 >, 10 :< 11, 9, 14, 3 >, 11 :< 10, 12, 4 >, 12 :< 13, 11, 5 >, 13 :< 8, 12, 14, 6 >, 14 :< 10, 13, 7 >\}$.

7.1.2 General construction of bg's

In what follows, we present a general construction for creating the first infinite families of broadcast graphs under the fully-adaptive model for $n = 6 \times 2^k$, 7×2^k, 9×2^k, and 10×2^k.

Lemma 7.1.2. *Consider a graph $G = (V, E)$ with n vertices, m edges, and $B_{fa}(G) = \tau$. It is always possible to construct a graph $G' = (V', E')$ with $2n$ vertices, $2m+n$ edges, and $B_{fa}(G') = \tau + 1$.*

Proof. First, make two copies of G and denote them by $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Also, assume that $|V_1| = |V_2| = n$, and $|E_1| = |E_2| = m$. Let $V_1 = \{x_1, x_2, \cdots, x_n\}$ and $V_2 = \{y_1, y_2, \cdots, y_n\}$, where y_i is the image of x_i; $i = 1, \cdots, n$. The vertices of G' consist of the union of those graphs: $V' = V_1 \cup V_2$. Also, the edges of G' comprise of the set of edges of G_1 and G_2 as well as n additional inter-layer edges: $E' = E_1 \cup E_2 \cup E_{\text{inter}}$, such that: $E_{\text{inter}} = \{(x_i, y_i) | \forall i : 1 \leq i \leq n\}$. Figure 7.2 portrays this construction.

Denote the broadcast scheme of graph G under the fully-adaptive model by σ such that $B^{\sigma}_{fa}(G) = \tau$. We are interested in making a broadcast scheme σ' for G' in a way that $B^{\sigma'}_{fa}(G') = \tau + 1$. The vertices of G' will preserve their broadcast scheme in G and append their corresponding vertex as the last vertex on their ordering. In other words, the ordering of vertex $x_i, 1 \leq i \leq n$ is as follows: $x_i :<$ within $G_1, y_i >$. Similarly, $\forall i; 1 \leq i \leq n : y_i :<$ within $G_2, x_i >$. These orderings imply that once a vertex gets informed, it will follow the broadcasting within its subgraph, either G_1 or G_2, and on its last call, it will inform its corresponding vertex. Suppose vertex u is the originator in graph G', which is located in partition $p \in \{1, 2\}$. Broadcasting from originator u in G' under the fully-adaptive model could be done in two phases:

- *Phase 1:* During the first τ time units, all vertices in partition p will be informed following σ' since each vertex will follow the same procedure as it had to follow within the smaller sub-graph G_p.
- *Phase 2:* At time $\tau + 1$, every vertex of partition p will make a call to their corresponding vertex in the other partition simultaneously.

Note that a vertex can finish its first phase sooner than time τ and send the message to its corresponding vertex at a time $t' < \tau$. However, this will not slow down the broadcasting process. Therefore, $B^{\sigma'}_{fa}(G') = \tau + 1$. □

We will use the following notation for expressing the result of Lemma 7.1.2:

Notation 7.1.1. $(G, n, m, \tau) \rightarrow (G', 2n, 2m + n, \tau + 1)$.

In which the arguments are the graph, the number of vertices, the number of edges, and the broadcast time of the graph under the fully adaptive model, respectively. Thereafter, using Notation 7.1.1, we will argue the following corollaries:

Corollary 7.1.2. *For any positive k:* $(G, n, m, \tau) \rightarrow (G', 2^k n, 2^k m + k2^{k-1}n, \tau + k)$.

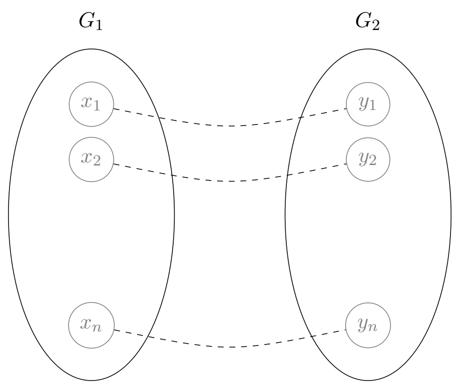


Figure 7.2: Construction of graph G'

Proof. The truth of this Corollary could be realized by repeating the procedure of Notation 7.1.1 for k times. Observe that the number of vertices doubles each time. Also, the number of edges by repeating this procedure for k times, m_k, could be understood by solving the following recursion: $m_k = 2m_{k-1} + n_{k-1}$, in which m_{k-1} refers to the number of edges by applying the construction for $k-1$ times, and n_{k-1} denotes the number of nodes by applying the construction for $k-1$ times. The base case of this recursion is $m_0 = m$ and $n_0 = n$. □

Corollary 7.1.3. *For any positive k, $(G, n, m, \lceil \log n \rceil) \rightarrow (G', 2^k n, 2^k m + k2^{k-1}n, \lceil \log n \rceil + k)$.*

Proof. This could be realized by replacing τ with $\lceil \log n \rceil$ in Corollary 7.1.2. Note that using this construction for a broadcast graph G on n vertices, the obtained graph G' remains a bg on $n \cdot 2^k$ vertices. □

An upper bound for the number of edges of a graph on $2^k n$ vertices could be realized by considering a complete graph on $2^k n$ vertices which have $\frac{2^k n \times (2^k n - 1)}{2} = 2^{k-1}(2^k n^2 - n)$ vertices. However, the graph G' presented in Corollary 7.1.3 has $2^{k-1}(2m + kn)$ edges. Therefore, for infinitely large values of n, as long as $m \in o(n^2)$, the construction for the graph G' will create a sufficiently sparse graph compared to the complete graph. This makes this construction efficient in terms of the cost associated with creating edges for a fixed value of n. Using Corollary 7.1.3, we will introduce some broadcast graphs for several values of n.

Proposition 7.1.1. *For any positive k:*

$(G, 6, 6, 3) \rightarrow (G', 6 \cdot 2^k, 6 \cdot 2^k + 6k2^{k-1}, 3 + k)$,

$(G, 7, 8, 3) \rightarrow (G', 7 \cdot 2^k, 8 \cdot 2^k + 7k2^{k-1}, 3 + k)$,

$(G, 9, 10, 4) \rightarrow (G', 9 \cdot 2^k, 10 \cdot 2^k + 9k2^{k-1}, 4 + k)$,

$(G, 10, 12, 4) \rightarrow (G', 10 \cdot 2^k, 12 \cdot 2^k + 10k2^{k-1}, 4 + k)$.

Proof. The proof of this Proposition directly follows from Corollary 7.1.3 and mbg's on $n = 6, 7, 9,$ and 10 vertices presented earlier in Figure 7.1. □

The following Theorem summarizes the presented results:

Theorem 7.1.1. *For any integer* $k = \lceil \log n \rceil \geq 4$*:*

$B^{(fa)}(n) = B^{(fa)}(2^{k-1} + 2^{k-4}) \leq \frac{n\lceil \log n \rceil}{2} - \frac{8n}{9}$,

$B^{(fa)}(n) = B^{(fa)}(2^{k-1} + 2^{k-3}) \leq \frac{n\lceil \log n \rceil}{2} - \frac{4n}{5}$,

$B^{(fa)}(n) = B^{(fa)}(2^{k-1} + 2^{k-2}) \leq \frac{n\lceil \log n \rceil}{2} - \frac{n}{2}$,

$B^{(fa)}(n) = B^{(fa)}(2^{k-1} + 2^{k-2} + 2^{k-3}) \leq \frac{n\lceil \log n \rceil}{2} - \frac{5n}{14}$.

Lastly, we prove the broadcast time of the hypercube under the fully-adaptive model.

Theorem 7.1.2. $B_{fa}(H_d) = d$

Proof. The proof will directly follow from Corollary 7.1.3 by choosing G as a graph on two vertices with one edge connecting them and repeating the procedure for $d - 1$ times: $(H_2, 2, 1, 1) \rightarrow (H_d, 2^d, d \cdot 2^{d-1}, d)$. The ordering of a vertex $\alpha = a_0 a_1 \cdots a_{d-1}$ with fully-adaptive broadcast time of d is as follows: $\alpha :< \alpha(d-1), \cdots, \alpha(1), \alpha(0) >$, where $\alpha(i)$ is the ith dimensional neighbour of α. □

Since there exists a broadcast scheme for any hypercube on 2^d vertices that achieves $B_{fa}(H_d) = d$, and using Lemma 7.1.1, one can conclude the following:

Corollary 7.1.4. *Hypercube* H_d *is an mbg on* 2^d *vertices under* $M = fa$*, and* $B^{(fa)}(2^k) = k \cdot 2^{k-1}$ *for any* $k \geq 1$.

7.2 Comparing broadcast time of various graphs

In Table 7.2, we summarize the known results for all graphs described earlier in Section 2.5. Some points should be highlighted regarding this table:

- The value of $B_{fa}(P_n)$ and $B_a(P_n)$ are established by using Theorem 5.2.1. The same is true for Star S_n under fully-adaptive and adaptive models.

- Any broadcast scheme yields the obvious lower bound of $B_{fa}(C_n) = B_a(C_n) = \lceil \frac{n}{2} \rceil$. Thus, we do not provide rigorous proof.

- The upper bound on $B_{fa}(K_n)$ and $B_a(K_n)$ is concluded from Theorem 5.1.1 and the upper bound provided in (Diks & Pelc, 1996).

- The value of $B_{cl}(K_{m\times n})$ is established in Theorem 6.1.2, while the upper bounds of this graph on universal lists are presented in Theorem 6.1.3.

- For complete $k-$ary tree, the classical broadcast time is reported in (Harutyunyan & Liestman, 1998) (with a different notation than us). The fully-adaptive and adaptive broadcast time is concluded based on Theorem 5.2.1. We also provided the non-trivial proof that gives $B_{na}(T_{k,h})$ in Theorem 6.1.1. The same is true for the Binomial tree, where we provided the proof for $B_{na}(T_d)$ in Proposition 6.1.1.

- The value of $B_{fa}(G_{m\times n})$ is established in Corollary 5.2.1. Also, the upper bound on $B_{fa}(T_{m\times n})$ is discussed in Theorem 5.2.2.

- The value of $B_{fa}(H_d)$ was studied in Theorem 7.1.2, while the upper bounds on non-adaptive and adaptive models for this graph are established based on the results given in (Harutyunyan & Liestman, 1998) and by using Lemma 6.2.1. For Cube Connected Cycle, we proved the fully-adaptive time in Theorem 5.2.3, whereas the non-adaptive and adaptive upper bounds are concluded from (Harutyunyan & Liestman, 1998) and Lemma 6.2.1.

- Also, the upper bounds on the universal lists broadcast time of Shuffle Exchange and De Bruijn networks are concluded from (Harutyunyan & Liestman, 1998) and by using Lemma 6.2.1.

- Finally, we could not find any results regarding the Harary networks, even for the classical model. However, we refer to (Bhabak & Harutyunyan, 2014) for a $\log \frac{k-2}{2}-$additive approximation algorithm for $H_{k,n}$.

Table 7.2: Broadcast time of well-known networks under models $M = \{cl, fa, a, na\}$.

Graph G	$B_{cl}(G)$	$B_{fa}(G)$	$B_a(G)$	$B_{na}(G)$
Path P_n	$n-1$, folklore	$n-1$	$n-1$	$\lceil \frac{3n}{2} \rceil - 2$, (Diks & Pelc, 1996)
Cycle C_n	$\lceil \frac{n}{2} \rceil$, folklore	$\lceil \frac{n}{2} \rceil$	$\lceil \frac{n}{2} \rceil$	$\lfloor \frac{2n}{3} \rfloor$, (Diks & Pelc, 1996)
Star S_n	$n-1$, folklore	$n-1$	$n-1$	n, (Diks & Pelc, 1996)
Complete graph K_n	$\lceil \log n \rceil$, folklore	$\leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil$	$\leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil$, (Diks & Pelc, 1996)	$\leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil$, (Diks & Pelc, 1996)
Complete Bipartite $K_{m\times n}$	$t_1 = \lceil \log n \rceil + 1 + \max\{\lceil \frac{m-2^{\lceil \log n \rceil}}{n} \rceil, 0\}$	$\leq t_1 + 3\lceil \sqrt{t_1} \rceil$	$\leq t_1 + 3\lceil \sqrt{t_1} \rceil$	$\leq t_1 + 3\lceil \sqrt{t_1} \rceil$
Complete $k-$ary tree $T_{k,h}$	$kh + h - 1$, (Harutyunyan & Liestman, 1998)	$kh + h - 1$	$kh + h - 1$	$kh + 2h - 1$
Binomial tree T_d	$2d - 1$, folklore	$2d - 1$	$2d - 1$	$3d - 2$
Grid $G_{m\times n}$	$m + n - 2$, (Farley & Hedetniemi, 1978)	$m + n - 2$	$m + n - 2$, (Diks & Pelc, 1996)	$m + n - 1$, (Diks & Pelc, 1996)
Tori $T_{m\times n}$	$\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor$, if m and n are even $\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 1$, otherwise (Farley & Hedetniemi, 1978)	$\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor$, if m and n are even $\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 1$, if only one of m and n is even $\leq \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 2$, otherwise	$\leq \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 3$, (Diks & Pelc, 1996)	$\leq \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 5$, (Harutyunyan & Taslakian, 2004)
Hypercube H_d	d, folklore	d	$\leq \frac{d(d-1)}{2} + 1$	$\leq \frac{d(d+1)}{2} + 1$
Cube Connected Cycle CCC_d	$\lceil \frac{5d}{2} \rceil - 1$, (Liestman & Peters, 1988)	$\lceil \frac{5d}{2} \rceil - 1$	$\leq 2\lceil \frac{5d}{2} \rceil - 1$	$\leq 3\lceil \frac{5d}{2} \rceil - 3$
Shuffle Exchange SE_d	$2d - 1$, (Hromkovič et al., 1993)	$\leq 4d - 1$	$\leq 4d - 1$	$\leq 6d - 3$
De Bruijn DB_d	$\leq \frac{3}{2}(d + 1)$, (Bermond & Peyrat, 1988) $\geq 1.3171d$, (Klasing et al., 1994)	$\leq 3d + 1$	$\leq 3d + 1$	$\leq 4d$
Harary graph $H_{k,n}$	?	?	?	?

Chapter 8

HUB-GA: A Heuristic for Universal lists Broadcasting using Genetic Algorithm

The problem of broadcasting with universal lists has several applications and could be applied to different networks, such as Software Defined Networks (SDNs). The SDN environment and its related algorithms have attracted both academia and industrial communities in recent years (Al-Jawad, Comşa, Shah, Gemikonakli, & Trestian, 2021; Binh & Duong, 2021; D. Li et al., 2017). For instance, the problem of broadcasting with universal lists has immediate applications in the update procedure of SDNs, which has shown to be highly effective on the performance of the networks and their QoS (D. Li et al., 2017; Sezer et al., 2013).

The emergence of Machine Learning has revolutionized research in almost all areas. In this chapter, we focus on the problem of broadcasting with universal lists and propose a reinforcement learning-based algorithm for this problem. Although there are several studies for this problem, our work differs from all in that we present the first heuristic for this problem. Our heuristic is based on the Genetic Algorithm, which is demonstrated to be genuinely useful for finding near-optimal solutions for problems with huge search spaces. The experimental results conducted in this chapter reveal that our approach is quite efficient while being tested under divergent circumstances.

The rest of this chapter is arranged as follows: In Section 8.1, the preliminaries of this work are studied, and we motivate the reader by providing an example. The related works concerning the genetic algorithm are discussed in Section 8.2. We propose our methodology in Section 8.3, and the numerical results are reported in Section 8.4.

8.1 Motivation and an Example

In universal lists broadcasting, once a vertex receives the message, it transmits it to its neighbors with respect to the fixed ordering given in the list. There are three sub-models defined using universal lists:

- **Non-adaptive model:** Once a vertex u receives the message, it will re-transmit it to all the vertices on its list, even if u has received it from some of its neighbors. The broadcast time of a graph G following this model is denoted by $B_{na}(G)$.
- **Adaptive model:** Once informed, a vertex u will send the message to its neighbors according to its list, but it will skip the neighbors from which it has received the message. The broadcast time of a graph G following this model is denoted by $B_a(G)$.
- **Fully adaptive model:** Once a vertex u gets informed, it will send the message to its neighbors according to its list, but it will skip all informed neighbors. The broadcast time of a graph G following this model is denoted by $B_{fa}(G)$.

There could be many unnecessary calls under the non-adaptive model, in which the message is returned to the sender. This model is the slowest among all while having the best space complexity, as the list is the only thing that should be maintained. Although the number of unnecessary calls drops in the adaptive model compared to the non-adaptive model, many calls might still be unnecessary since only the senders are avoided under this model. Hence, a vertex u may transmit the message to one of its neighbors v, which is already informed but has not yet transmitted the message to u. Furthermore, each vertex requires additional storage to keep track of the nodes from which it has received the message. Lastly, a vertex skips all of its informed neighbors in the Fully-adaptive model. This accelerates the broadcast process compared to the adaptive model. Similar to the classical model, no unnecessary call is made following this model. The space complexity, however, increases compared with the adaptive model since a vertex has to be aware of the state of its neighbors. Earlier, we proved that:

$$B_{cl}(G) \leq B_{fa}(G) \leq B_a(G) \leq B_{na}(G) \tag{28}$$

The relations in Equation (28) are reversed considering the required space. In summary, the choice of a suitable model heavily depends on the available resources and the requirements of the network.

Consider graph $G = (V, E)$. A broadcast scheme for the non-adaptive, adaptive, or fully adaptive model can be viewed as a matrix $\sigma_{n\times\Delta}$, where row i of σ corresponds to an ordering of the neighbors of vertex v_i. Assuming this vertex has degree d_i, the cells $\sigma_{[i][d_i+1]}, \sigma_{[i][d_i+2]}, \cdots, \sigma_{[i][\Delta]}$ will be `Null`, where $\Delta = \max\{d_i : 1 \leq i \leq n\}$. Also, denote all possible schemes for a graph G by $\Sigma_{(G)}$. When it is clear from the context, we may omit the subscript (G).

Let M be one of the three models using universal lists ($M \in \{na, a, fa\}$) and fix a graph G. For any broadcast scheme $\sigma \in \Sigma$, $B^\sigma_M(u, G)$ is the time steps needed to inform all the vertices in G from the source u while following the scheme σ under model M. Moreover, the broadcast time of a graph G under model M with scheme σ is defined as the maximum $B^\sigma_M(u, G)$ over all possible originators. Finally, $B_M(G)$ is the minimum $B^\sigma_M(G)$ over all possible schemes:

$$\begin{aligned} B^\sigma_M(G) &= \max_{u\in V}\{B^\sigma_M(u, G)\} \\ B_M(G) &= \min_{\sigma\in\Sigma}\{B^\sigma_M(G)\} \end{aligned} \tag{29}$$

8.1.1 An example

Table 8.1 Shows an arbitrary σ for the graph given in Figure 8.1-a. Suppose vertex 3 is the originator. Now, the broadcast processes starting from this vertex under all three models are described:

- **Fully-adaptive:** At time $t = 1$, vertex 3 sends the message to vertex 2 according to its list. At time $t = 2$, vertex 3 sends the message to vertex 4, and vertex 2 (by skipping the informed vertex 3) will send the message to vertex 1. At time $t = 3$, the remaining vertices on the list of vertices 1 and 2 are already informed; thus, vertices 1 and 2 will remain idle. Also, vertex

3 will skip vertex 1 (since it is informed) and send the message to vertex 5, whereas vertex 4 will skip vertices 3 and 2 from its list and send the message to vertex 6. Since all 6 vertices are informed, $B^{\sigma}_{fa}(G, 3) = 3$. See Figure 8.1-b.

- **Adaptive:** The broadcast process is the same for time units $t = 1, 2$. However, at time $t = 3$, vertex 1 sends the message to vertex 3, while vertex 2 sends the message to vertex 4 because vertex 2 has not received the message from vertex 4. Also, vertex 3 sends the message to vertex 1 for the same reason, whereas vertex 4 is the only vertex that informs an already uninformed vertex 6. Since vertex 5 is still uninformed, the process must continue for another time unit $t = 4$, in which vertex 3 and 6 hit the same target (vertex 5) which ends the process: $B^{\sigma}_{a}(G, 3) = 4$. See Figure 8.1-c.

- **Non-adaptive:** This model is the slowest model yet the easiest to follow because no vertex is being skipped. Therefore, once a vertex gets informed at time t, it will send the message to its neighbors at time $t + 1, t + 2, \cdots$, following its list given in σ. For instance, since vertex 3 is the originator who has the message at time $t = 0$, it will pass the message to vertices 2, 4, 1, and 5 at time units $t = 1, 2, 3, 4$, respectively. Also, since vertex 2 is informed at time $t = 1$, it will send the message to vertex 3,1, and 4 during time units $t = 2, 3, 4$, respectively. Following this process for other vertices is the same, therefore is omitted. Eventually, this process ends at time unit 5: $B^{\sigma}_{na}(G, 3) = 5$. See Figure 8.1-d.

Two interesting points could be emphasized from this example: Firstly, while a broadcast scheme σ might be optimal for a particular model, it could be inefficient under another model. For instance, Table 8.1 is optimal under model *fa* and vertex 3 as the originator, whereas it is not optimal under model *na* and the same originator. Secondly, a broadcast scheme could be tailored for a certain originator, whereas it is not efficient once other vertices are selected as the message originator. For example, following Table 8.1 from vertex 6 as the originator, the fully-adaptive broadcasting will be finished in four time units even though it is possible to come up with another scheme that reduces this time to 3. The tough task is to design a scheme σ' that yields a broadcast time of 3 from not only vertex 6 but all vertices of this graph under the fully-adaptive model. This broadcast scheme is given in Table 8.2. Following this optimal scheme $B^{\sigma'}_{fa}(G) = 3$. Also,

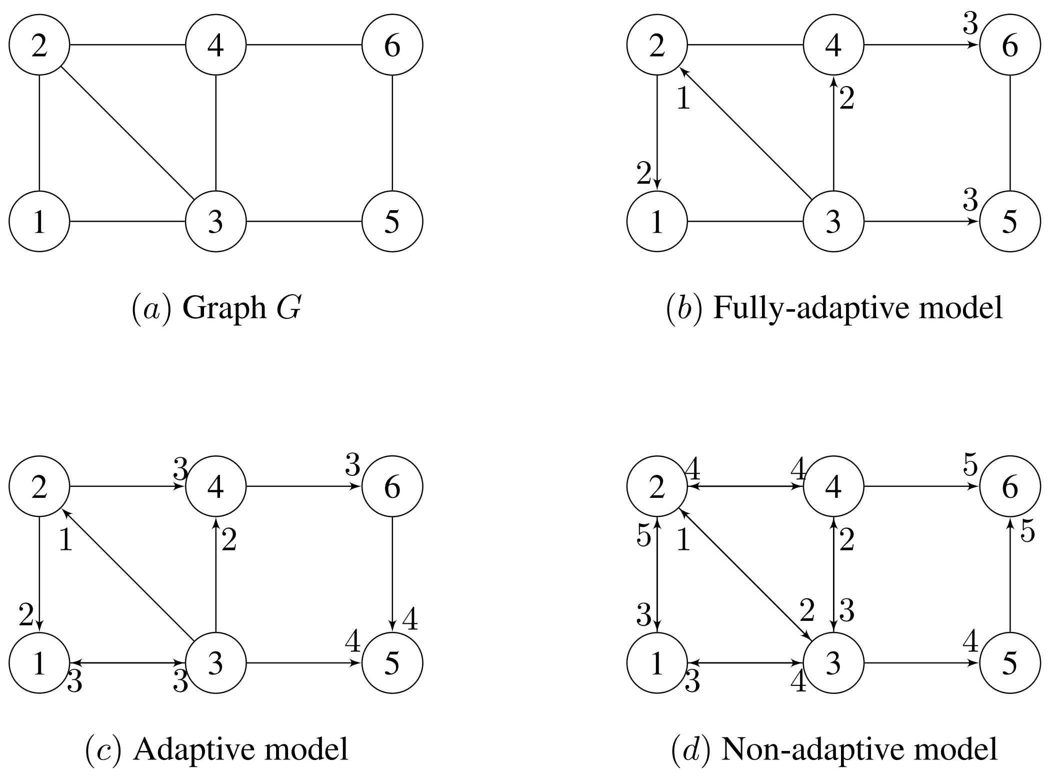


Figure 8.1: a) Graph G, b) Broadcast Scheme of G Under Fully Adaptive Model: $B^{\sigma}_{fa}(G, 3) = 3$, c) Broadcast Scheme of G Under Adaptive Model: $B^{\sigma}_{a}(G, 3) = 4$, d) Broadcast Scheme of G Under Non-Adaptive Model: $B^{\sigma}_{na}(G, 3) = 5$

Table 8.3 gives an optimal broadcast scheme under the adaptive model with $B^{\sigma'}_{a}(G) = 3$, whereas Table 8.4 is an optimal scheme under the non-adaptive model with $B^{\sigma'}_{na}(G) = 4$ (achieving the non-adaptive broadcast time of 3 for this graph is impossible).

Table 8.1: An Ordering of Vertices for the Graph of Figure 8.1

Sender	Ordering of receivers			
1	3	2	NULL	NULL
2	3	1	4	NULL
3	2	4	1	5
4	3	2	6	NULL
5	6	3	NULL	NULL
6	5	4	NULL	NULL

It is evident that guessing an optimal scheme out of many possible solutions is nearly impossible, even for a small graph. Besides, our results demonstrate that simple heuristics, such as sorting the lists based on the degree, are also ineffective. In this study, we propose a heuristic based on the

Table 8.2: An optimal broadcast scheme for the Graph given in Figure 8.1 under fully-adaptive model

Sender	Ordering of receivers			
1	2	3	NULL	NULL
2	1	4	3	NULL
3	4	1	2	5
4	6	3	2	NULL
5	6	3	NULL	NULL
6	4	5	NULL	NULL

Table 8.3: An optimal broadcast scheme for the Graph given in Figure 8.1 under adaptive model

Sender	Ordering of receivers			
1	3	2	NULL	NULL
2	4	3	1	NULL
3	5	1	2	4
4	2	3	6	NULL
5	3	6	NULL	NULL
6	5	4	NULL	NULL

Genetic algorithm that tackles this problem.

8.2 Related works on Genetic Algorithm

Genetic Algorithm (GA) has attracted an enormous amount of attention from researchers after being proposed by J. H. Holland (Holland, 1992). The studies in this domain could be divided into several categories.

Enhancing the algorithm: Several researchers have tried to enhance the GA itself by proposing different approaches for various operations of GA, such as Crossover (Louis & Rawlins, 1991; Semenkin & Semenkina, 2012; Tsutsui, Yamamura, & Higuchi, 1999), Mutation (Ahn & Ramakrishna, 2003), Selection (Fang & Li, 2010; Goldberg, 1990; Hutter, 2002; Ishibuchi & Yamamoto, 2004; R. Kumar, 2012; Miller & Goldberg, 1995), or the Stopping Criteria (Safe, Carballido, Ponzoni, & Brignole, 2004). For survey papers, we refer to (Goldberg & Deb, 1991; Grefenstette, 1993; Katoch, Chauhan, & Kumar, 2021; Mirjalili, 2019; Srinivas & Patnaik, 1994; Zbigniew, 1996).

Graph-related problems: GA helped researchers to develop heuristics for different problems. As far as graph-related problems are concerned, the graph coloring problem and some solutions

Table 8.4: An optimal broadcast scheme for the Graph given in Figure 8.1 under non-adaptive model

Sender	Ordering of receivers			
1	2	3	NULL	NULL
2	4	3	1	NULL
3	1	5	2	4
4	3	6	2	NULL
5	6	3	NULL	NULL
6	4	5	NULL	NULL

working with GA are discussed in (Assi, Halawi, & Haraty, 2018; Marappan & Sethumadhavan, 2018), while some algorithms for the graph partitioning with GA are proposed in (Maini, Mehrotra, Mohan, & Ranka, 1994; Talbi & Bessiere, 1991). Several algorithms for Max-cut (Y.-H. Kim, Yoon, & Geem, 2019), Max clique (Marchiori, 1998; Pinto, Ribeiro, Rosseti, & Plastino, 2018), and Max-Cut clique (Fortez, Robledo, Romero, & Viera, 2020) problems are proposed in the literature. Palmer and Kershenbaum (Palmer & Kershenbaum, 1994) came up with a novel idea for representing trees in GA and showed that the proposed encoding could be effective for several problems.

Routing: As mentioned in (Lambora, Gupta, & Chopra, 2019), GA has been used in various network routing protocols. In particular, in (Selvanathan & Tee, 2003), a GA approach for the shortest path problem under two different settings has been proposed. Also in (Shanmugasundaram, Sushita, Kumar, & Ganesh, 2019), the network design problem with the goal of optimizing vehicle travel distance has been studied with GA, while Unmanned aerial vehicle (UAV) networks (Wen et al., 2022), and energy-efficient resource allocation (Jiao & Joe, 2016) have been studied using GA. Digital Data Service (DDS), which is a famous communication service, is studied in (C.-H. Chu, Premkumar, & Chou, 2000) where the authors proposed a GA for the Steiner-tree problem that is tightly connected to the design of DDS networks.

In (Muruganantham & El-Ocla, 2020) the performance of several routing algorithms in Wireless Sensor Networks (WSNs) are compared, such as GA, Dijkstra algorithm, Ad hoc On-Demand Distance Vector (AODV), GA-based AODV Routing (GA-AODV), grade diffusion (GD) algorithm, directed diffusion algorithm and GA combined with the GD algorithm. They have also studied the performance of those algorithms in the presence of faulty nodes and found out that combining GA with other algorithms is usually useful. In (Mazaideh & Levendovszky, 2021), efficient data transmission through WSNs with multiple objective genetic algorithm is studied by proposing a compressive sensing-based algorithm. Multicast routing problem and an algorithm based on GA is suggested in (Sun & Li, 2006). A dynamic source routing protocol in Mobile ad hoc network (MANET) is suggested in (D.-g. Zhang, Liu, Liu, Zhang, & Cui, 2018), which utilizes genetic algorithm-bacterial foraging optimization (GA-BFO). The routing problem in MANETs is also discussed in (Bhardwaj & El-Ocla, 2020) using GA, considering energy consumption as one of the foremost vital limitations in MANETs. Furthermore, the effectiveness of GA-based routing solutions in Vehicular Ad-hoc networks (VANETs) is discussed in (A. Kumar, Dadheech, Kumari, & Singh, 2019). Also, QoS routing using GA in VANETs (G. Zhang, Wu, Duan, & Huang, 2018) and WSNs with applications in smart grids (Baroudi, Bin-Yahya, Alshammari, & Yaqoub, 2019) is a well-studied problem in the literature.

In (Bhola, Soni, & Cheema, 2020), the goal is to enhance energy efficiency with the lifespan of sensor nodes. They proposed an energy-effective routing protocol, low energy adaptive clustering hierarchy (LEACH) in addition to an optimization GA. The authors of (Wang, Zhang, Yang, & Vajdi, 2018) proposed a GA for routing in WSNs with the aim of minimizing energy consumption. The problem of routing in energy harvesting-wireless sensor networks (EH-WSN) has been studied in (Wu & Liu, 2013) using GA. Prolonging the lifetime of nodes by reducing energy consumption in WSNs is studied in (Rezaeipanah, Nazari, & Ahmadi, 2019) where GA has been used for the routing algorithm.

Broadcasting: In the area of broadcasting, Hoelting et al. proposed a GA for the problem of Minimum Broadcast Time (MBT) (Hoelting, Schoenefeld, & Wainwright, 1996). They tested their algorithm for random graphs on 10 to 500 nodes, as well as three contrived sets of networks on 40, 80, and 120 nodes. They compared their results with the Approximation Matching (AM) algorithm presented in (Scheuermann & Wu, 1984), and claimed that their algorithm outperforms AM, particularly considering the contrived networks. Moreover, Hasson and Sipper (Hasson & Sipper, 2004) proposed ACS: an Ant Colony System for the MBT problem. They compared the performance of their algorithm with that of (Hoelting et al., 1996) and AM (Scheuermann & Wu, 1984) for random graphs on 15 to 250 nodes and edge probability in the range of $(0.05 - 0.1)$. In most cases, the achieved broadcast time was better or the same, while the running time was enhanced compared to both algorithms.

More recently, Lima et al. (Lima, Aquino, Nogueira, & Pinheiro, 2022) proposed a metaheuristic algorithm for the MBT problem with GA, namely BRKGA. They suggested various versions of their algorithm while combining it with different decoders, such as First Receive First Send (FRFS) or an Integer Linear Programming (ILP) method. They compared their results with various methods such as Tree Block (de Sousa et al., 2018), NTBA (Harutyunyan & Wang, 2010), NEWH (Harutyunyan & Jimborean, 2014), and ACS (Hasson & Sipper, 2004) for several graph families of up to 1024 nodes. Their results show that BRKGA is able to outperform all counterpart heuristics for the MBT problem, while it can also be an alternative for larger networks where the exact methods cannot be applied. Finally, we should point out that the goal of (Hasson & Sipper, 2004; Hoelting et al., 1996; Lima et al., 2022) is to minimize the classical broadcast time of $\underline{\text{a particular vertex}}$ under the classical model, not to optimize the broadcast time of all network members at the same time.

8.3 Methodology

In this section, we propose a novel solution to the problem of broadcasting with universal lists. We look at this problem as follows: Assuming Σ as the search space, our goal is to find a $\sigma \in \Sigma$ that minimizes an objective function, i.e., the broadcast time. In the following proposition, it is argued that the size of the search space skyrockets as the graph size grows. Subsequently, a thorough exploration of state space is impossible.

Proposition 8.3.1. *For a graph G on n vertices, where the degree of vertex i is d_i, the size of search space for the problem of broadcasting with universal list is as follows:*

$$|\Sigma_{(G)}| = \prod_{i=1}^{n} \sum_{j=0}^{d_i} (\binom{d_i}{j} \times j!) \tag{30}$$

Proof. Note that, as mentioned in (Diks & Pelc, 1996), a solution to the problem of broadcasting with universal lists may include some neighbors of a particular vertex, not all of them, necessarily.

For a vertex i with d_i neighbors, a valid universal list may contain any number of its neighbors ranging from 0 to d_i. If it includes j neighbors, there are $\binom{d_i}{j}$ different ways to select those neighbors, while in each case, the selected neighbors may be arranged in $j!$ different ways. Also, this is true for each vertex $1, 2, \cdots, n$. $\square$

Observe that the function given in Equation (30) grows at least as fast as $\Omega\Big((\delta!)^n\Big)$ when δ is the minimum degree of the graph, that is, $|\Sigma_{(G)}|$ is at least exponential. The complexity of the same function and its importance are studied in (Morosan, 2006) under the classical model. All in all, since the search space size is exponential, evolutionary computing could be useful in finding an optimal solution out of many possible solutions with a high probability (Zbigniew, 1996). Genetic algorithm is a particular class of evolutionary algorithms that is effective in finding optimal solutions for complex problems in various domains such as biology, engineering, computer science, and social science.

Genetic Algorithm (GA), introduced by J. H. Holland (Holland, 1992), is an optimization algorithm. It is a population-based search algorithm (Katoch et al., 2021) that uses the idea of survival of the fittest (Zbigniew, 1996), originally inspired by the Darwinian theory of evolution (Grefenstette, 1993). In this algorithm, every solution to the problem at hand corresponds to a chromosome, while every parameter is a gene (Mirjalili, 2019). The fitness of each individual is evaluated with a fitness function. To improve the quality of solutions, the best solutions are selected for reproduction using two primary operations of GA: Crossover and Mutation. GA tries to find a suitable solution by repeating this process over multiple generations.

In this study, we propose `HUB-GA`: a $\underline{\text{H}}$euristic for $\underline{\text{U}}$niversal list model of $\underline{\text{B}}$roadcasting with $\underline{\text{G}}$enetic $\underline{\text{A}}$lgorithm. The general routine of `HUB-GA` is given in Algorithm 6, whereas Figure 8.2 portrays how this framework is tailored according to our problem. Note that the most crucial task for developing a GA heuristic for a problem is to encode the problem properties into chromosomes and genes (Selvanathan & Tee, 2003; Sun & Li, 2006). In the rest of this section, the detail of the proposed algorithm is explained for the problem at hand.

Algorithm 6 HUB-GA

```
1: Generate random population;
2: Calculate fitness score;
3: while not converged do
4:     Crossover;
5:     Mutation;
6:     Calculate fitness score;
7:     Acceptance;
8: end while
9: return The best chromosome
```

8.3.1 Genes, Chromosomes, and Population

Consider a graph G with n vertices and denote the degree of vertex i by d_i. An arbitrary ordering of the neighbors of vertex i represents a *gene* i, $g^{(i)}$, with size at most d_i. A *chromosome* is a collection of n genes: $g^{(1)}, g^{(2)}, \cdots, g^{(n)}$, each corresponding to an ordering for a particular vertex. Table 8.1 portrays an arbitrary chromosome for the graph given in Figure 8.1-a, where row i of the matrix corresponds to gene $g^{(i)}$. Using our notation, a chromosome is a matrix σ with n rows (or n genes) and Δ columns. We need to stress that a chromosome is different from an adjacency list of the graph. This is because a gene does not necessarily include all neighbors of a particular vertex, as opposed to the adjacency list.

In GA, a chromosome is a possible solution for the problem, meaning that any $\sigma \in \Sigma$ may be an optimal broadcast scheme. However, guessing the correct solution out of too many possible solutions is nearly impossible. Subsequently, the first step of HUB-GA is generating several solutions randomly, called the first *population*. The main goal of the initialization step is to distribute the solutions in the search space as evenly as possible to increase the diversity of the population and have a better chance of finding promising regions (Mirjalili, 2019). Consequently, the bigger the size of the first generation, the higher the chance of finding a near-optimal solution in early iterations. The computational cost, however, surges as the size of the population grows. Therefore, choosing a reasonable size for the first population is a trade-off between cost and accuracy. Later in Experiment 1, we study the effect of population size, or $|p|$, on the performance of our algorithm.

Finally, choosing a suitable encoding for genes, chromosomes, and population is crucial since they have to abide by some fundamental rules. For instance, a gene has to be mutable. Therefore, having modified the content of a gene, another valid solution must be generated. Moreover, creating new and unseen solutions must be possible using two different chromosomes. Also, as mentioned in (Palmer & Kershenbaum, 1994), it should be easy to go back and forth between the graph's encoding and the graph itself in a more conventional form suitable for evaluating the fitness function (the encoding/decoding process). Lastly, the encoding should possess locality; small changes in the encoding should result in small changes in the fitness score (Palmer & Kershenbaum, 1994). This will help GA to function more effectively. This study's novel encoding allows us to respect these ground rules without violating the problem's definitions.

8.3.2 Fitness Function

The key element of any GA-based heuristic for a problem is the definition of its fitness function. The fitness function, $f(\sigma)$, evaluates the fitness of a chromosome σ, and it must be designed in a way that it evaluates the optimal solution as the best one. The ultimate objective of GA is to find a chromosome σ which minimizes/maximizes a predefined fitness function $f(\sigma)$. We have considered two fitness functions in this study. In both cases, a better chromosome has a smaller fitness score. Hence, the ultimate objective is to minimize the fitness function.

Broadcast time

For the first case, the fitness function is considered to be the broadcast time of the graph using σ as the universal list:

$$f_1(\sigma) = \max_{u \in V(G)} \{B_M^{\sigma}(u, G)\} = B_M^{\sigma}(G) \tag{31}$$

To evaluate $f_1(\sigma)$, the broadcast process is simulated starting from every originator u, using σ as the universal list under a model M. No need to mention that this is done in parallel for each originator. Then, the maximum broadcast time over every originator is considered as $f_1(\sigma)$. Therefore, in this case, the aim is to minimize the maximum broadcast time. Eventually, we expect to find a scheme in which, starting from any originator, the broadcast time is quite short. Ideally, $f_1(\sigma)$ approaches $B_M(G)$, since $B_M(G) = \min_{\sigma \in \Sigma}\{B_M^{\sigma}(G)\}$.

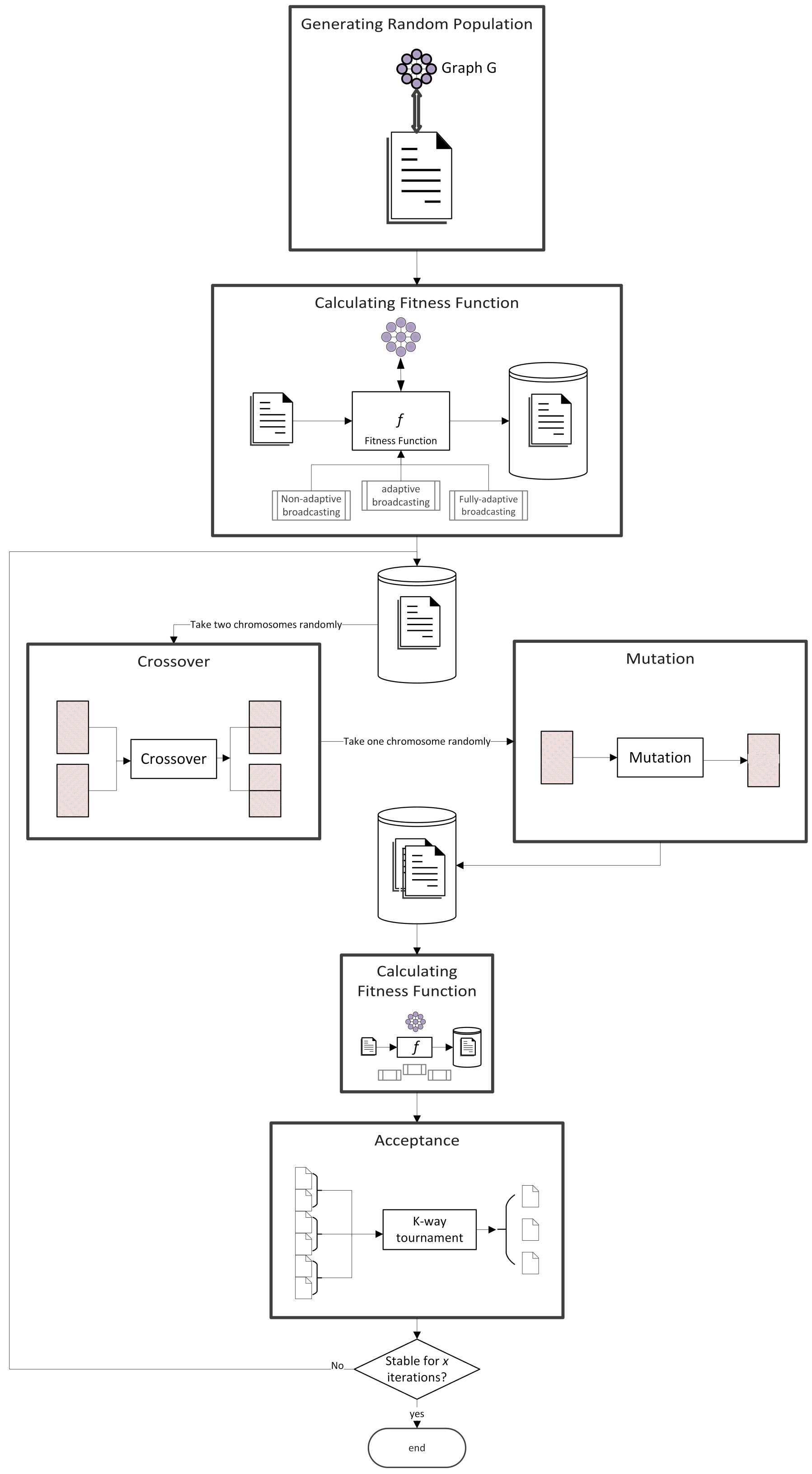

Generating Random Population
Graph G
Calculating Fitness Function
f
Fitness Function
Non-adaptive broadcasting
adaptive broadcasting
Fully-adaptive broadcasting
Take two chromosomes randomly
Crossover
Crossover
Take one chromosome randomly
Mutation
Mutation
Calculating Fitness Function
f
Acceptance
K-way tournament
No
Stable for x iterations?
yes
end

Figure 8.2: A schema of our methodology

The reason for defining this function is quite apparent. In recent years, finding the broadcast time of a particular family of graphs has been an ongoing research question (Bhabak & Harutyunyan, 2022; Slater et al., 1981), particularly under the universal list model (Diks & Pelc, 1996; Harutyunyan et al., 2011; Harutyunyan & Taslakian, 2004; J.-H. Kim & Chwa, 2005; Rosenthal & Scheuermann, 1987). Using the proposed algorithm, we can tackle this problem in two ways: Firstly, our algorithm generates the actual broadcast scheme that minimizes its fitness function. The scheme could be used as a foundation for proposing general upper bounds on the broadcast time of a particular family of graphs. Secondly, the results provided by this algorithm could question the reachability of some lower bounds. For instance, based on our experimental results, we conjecture that the broadcast time of a Complete Graph K_n on n vertices under the fully-adaptive model should be strictly greater than that of the classical model. This idea could even be proved by generating all possible schemes for a particular K_n and calculating their f_1 fitness score, which is out of the scope of this study. On the other hand, by using f_1 as the fitness function, the network manager will be able to minimize the maximum broadcast time of any network entity. This has shown to be helpful in different applications such as wireless sensor networks (Shang, Wan, & Hu, 2010), industry 4.0 (Hocaoğlu & Genç, 2019), video packet distribution in vehicular ad-hoc networks (Salkuyeh & Abolhassani, 2018), robotics (Bucantanschi, Hoffmann, Hutson, & Kretchmar, 2007), satellite link topology (X. Chu & Chen, 2018), and age of information performance (Liu, Huang, Li, & Ji, 2021).

Average broadcast time

In the second case, the average broadcast time of the network is used for the fitness function:

$$f_2(\sigma) = \frac{\sum_{u \in V(G)} B_M^{\sigma}(u, G)}{n} \tag{32}$$

In which n is the number of vertices of graph G. Evaluating $f_2(\sigma)$ is quite similar to that of $f_1(\sigma)$, except that instead of taking the maximum broadcast time of each originator, we use the average broadcast time. Using this fitness score, we aim to find a scheme σ for which the behavior of all originators is fairly optimal in terms of broadcasting under model M.

The second function's definition fits this study's environment even better. To illustrate, by making a slight change in a chromosome σ, the value associated with $f_2(\sigma)$ is also expected to change as opposed to $f_1(\sigma)$, which is likely to remain the same. Previously referred to as locality, this vital feature could help create a more discoverable search space. Subsequently, the GA is anticipated to perform better in minimizing f_2 in general.

As an application concerning f_2, consider a communication network containing a clique and several paths attached to its nodes. In this network, the furthest nodes from the clique cannot inform all members quickly, regardless of the scheme. Using f_1 for this network means paying attention only to those at the furthest distance from the clique members. Consequently, the broadcast time of several vertices will not be minimized, even though it could be essential for the network provider. Using f_2, on the other hand, will result in a scheme in which the behavior of every originator is objectively optimal.

8.3.3 Crossover

There are two basic GA operators; the first one is Crossover, where two chromosomes are selected as the parents (*selection* phase), and then two children (called *offsprings*) are generated by *crossover*. Each of these steps is described hereafter.

In nature, the fittest individuals are more likely to get food and mate (Mirjalili, 2019). Inspired by this fact, the GA gives a higher chance to fitter chromosomes to be selected as the parents. There are several solutions proposed for this phase, a.k.a. *selection*. For instance, one solution is to use the Roulette Wheel method (Goldberg & Deb, 1991), where the chance of selecting a chromosome is proportional to its fitness score. Other well-known selection operations include, but are not limited to, Boltzmann selection (Goldberg, 1990), Rank selection (R. Kumar, 2012), Fuzzy selection (Ishibuchi & Yamamoto, 2004), and Fitness uniform selection (Hutter, 2002).

In this study, we used the famous $K-$way Tournament selection method (Miller & Goldberg, 1995). In this method, K individuals are selected from the population randomly. Then, the fittest chromosome, according to its fitness function, will be chosen for the role of the first parent. The same procedure is repeated for selecting the second parent. This method is genuinely valuable, mainly when the fitness score is a minimization function. Moreover, this method is very efficient for parallel computing (Fang & Li, 2010). In this chapter, K is fixed to 4 as a commonly used value (Fang & Li, 2010; Miller & Goldberg, 1995).

Once parent 1 (p_1) and parent 2 (p_2) are selected, a random integer is chosen from the range $[1, n)$ as the crossover point, denoted by c_p. Afterward, two new off-springs (c_1 and c_2) are generated as follows:

$$\begin{aligned} c_1 &= g^{(1)}(p_1), \cdots, g^{(c_p)}(p_1), g^{(c_p+1)}(p_2), \cdots, g^{(n)}(p_2) \\ c_2 &= g^{(1)}(p_2), \cdots, g^{(c_p)}(p_2), g^{(c_p+1)}(p_1), \cdots, g^{(n)}(p_1) \end{aligned} \tag{33}$$

In which $g^{(i)}(p_j)$ is the ith gene of parent j. Indeed, the first child contains the first c_p genes of p_1, while other genes come from p_2. This will be the other way around for the second child. This method is called single-point crossover, one of the most common methods in addition to the double-point method (Srinivas & Patnaik, 1994). To name a few other methods, Three parents crossover (Tsutsui et al., 1999), Uniform crossover (Semenkin & Semenkina, 2012), and Masked crossover (Louis & Rawlins, 1991) could be noted.

We keep making new off-springs until the initial size of the population is doubled. Note that off-springs still are valid solutions since no new vertex is added to the neighborhood of a sender. However, their fitness function still needed to be calculated in the next iteration.

8.3.4 Mutation

The second operator of GA is mutation, in which a gene of an offspring is changed randomly with a small probability. Mutation preserves population diversity by introducing another level of randomness (Mirjalili, 2019). Also, this operator prevents the solutions from becoming similar and increases the probability of avoiding local solutions in the GA (Mirjalili, 2019).

In our algorithm, the ordering of a gene is shuffled when it is supposed to go over mutation. Although mutation prevents GA from falling into a local optimum, there is a big chance of losing the best chromosome. Therefore, we performed two actions for mutation:

(1) **Decreasing mutation probability:** The chance of performing mutation reduces over time.

This is because, in early iterations, the goal is to explore the state space as much as possible, which is achieved by mutation. However, during last iterations, the algorithm probably converged with a relatively good solution. Therefore, mutation cannot be useful anymore.

(2) **Elitism:** When a gene of a chromosome is changed by mutation, the best solution is likely to be lost. Using Elitism (Ahn & Ramakrishna, 2003), a copy of the original individual is maintained before changing it. This copy will be carried out to the next generation, being treated as a normal chromosome. The goal is to prevent such solutions (elites) from being degraded. This method may accelerate the convergence of GA dramatically (Ahn & Ramakrishna, 2003). Also, as mentioned in (Mirjalili, 2019), the reliability of GA heavily depends on the process of maintaining the best solutions in each generation. Using Elitism, the chance of maintaining such solutions gets higher.

In our implementation, at early iterations, each gene could go over mutation with a 1% chance, and this number decreases smoothly over time. When a mutation is supposed to happen, a copy of the original chromosome is kept in the population.

Figure 8.3 shows the three main operators of HUB−GA, i.e. Selection, Crossover, and Mutation. In this Figure, Parent 1 (p_1) and Parent 2 (p_2) are the same chromosomes as the ones given in Table 8.1 and Table 8.2 for the Graph displayed in Figure 8.1-a. Assume p_1 and p_2 are selected through a 4-Way tournament among other candidates. By performing crossover with $c_p = 2$ on p_1 and p_2, two new offsprings, c_1 and c_2, are generated. Each gene of c_1 and c_2 may go through mutation with a small probability, resulting in a random shuffle in $g^{(3)}$ of c_1.

8.3.5 Acceptance

Having generated several off-springs using crossover and mutation, the population size escalates. One possible solution to keep the current generation manageable with limited resources is to retain the original population size by allowing a fixed number of chromosomes to survive into the next generation.

In order to do so, we used a 4-way tournament as described earlier in section 8.3.3. Thus, as long as the size of the next generation has not reached the original size, four random chromosomes

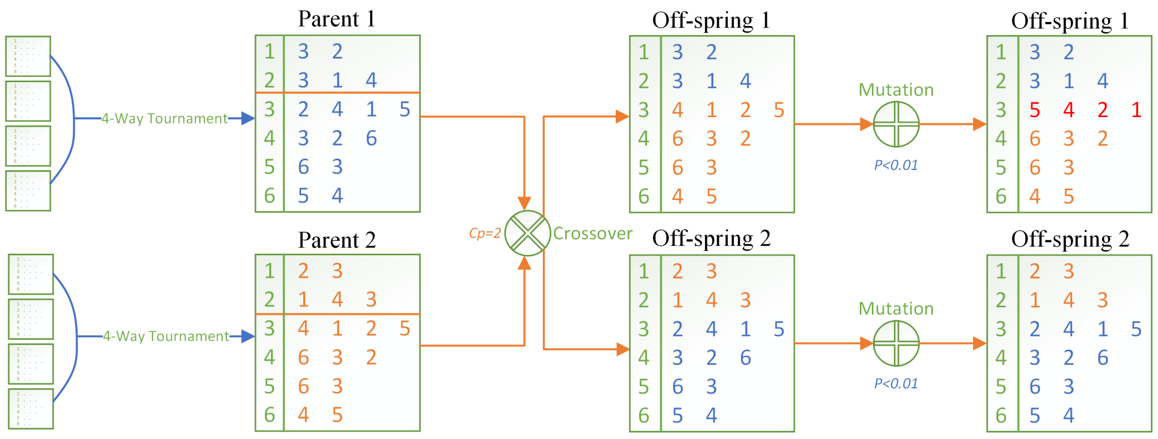


Figure 8.3: The process of Selection, Crossover, and Mutation in HUB−GA

are selected from the current generation, and the one with a better fitness score will survive into the next generation. This method gives a higher chance to the fittest individuals to survive, while any chromosome could be carried out to the next generation.

8.3.6 Stopping criterion

Several conditions are discussed in the literature regarding the best time to terminate the execution of GA (Safe et al., 2004; Zbigniew, 1996). Each criterion has advantages and disadvantages. For instance, the most trivial way is to stop exploring when a fixed number of generations have been generated or a predefined time limit is exceeded. Although the idea is simple and easy to implement, choosing a number for this purpose is not straightforward. Another well-known solution is to stop the execution when the lower bound of the problem is met. Utilizing this condition could be quite effective in many problems. In the case of broadcasting, however, there are two obstacles. Firstly, calculating the lower bound on $B_M(G)$ could be as difficult as guessing $B_{cl}(G)$ (See Eq. (28)), which is an NP-Hard problem (Garey & Johnson, 1983). Secondly, to the best of our knowledge, there is no solution to evaluate the lower bound's reachability for a particular graph. Therefore, if the lower bound is unachievable, the GA never terminates.

In this study, we employed an adaptive stopping criteria. The execution of HUB−GA terminates if, after S_t iterations, the fitness score of the fittest individual does not change. Thus, the algorithm can avoid unnecessary reproductions once a local (and possibly global) optimum has been reached.

Nonetheless, choosing the parameter S_t remains a confusing puzzle. Therefore, in Experiment 1, we studied the impact of changing parameter S_t, a.k.a the stability variable, on the performance of our algorithm. Once the stopping criterion is met, the best chromosome (solution) from the current generation and its fitness score are returned as the final answer.

8.3.7 Remarks

The advantages of our method are as follows:

(1) **The first heuristic for this problem:** As mentioned in Section 8.2, there are several heuristics for classical broadcasting (Beier & Sibeyn, 2000; Harutyunyan & Shao, 2006), even some researchers consider GA (Hoelting et al., 1996; Lima et al., 2022), or other evolutionary algorithms (Hasson & Sipper, 2004). However, to the best of our knowledge, there is no similar study on the problem of broadcasting with universal lists.

(2) **Working for arbitrary graphs:** the proposed framework takes an arbitrary graph as the input and finds a nearly optimal broadcast scheme for that particular graph. This is a tremendous advantage since all upper bounds provided in the literature are tailored for a specific family of networks (Diks & Pelc, 1996; Harutyunyan et al., 2011; Harutyunyan & Taslakian, 2004; J.-H. Kim & Chwa, 2005).

(3) **Working for any model under universal lists:** our heuristic could minimize its objective function (either f_1 or f_2) under all three models: non-adaptive, adaptive, or fully adaptive. To the best of our knowledge, this novelty separates `HUB-GA` from several related works considering a specific model; either classical (Bar-Noy et al., 1998; Beier & Sibeyn, 2000; Bhabak & Harutyunyan, 2022; Harutyunyan & Shao, 2006; Hasson & Sipper, 2004; Hoelting et al., 1996; Kortsarz & Peleg, 1995; Lima et al., 2022; Ravi, 1994), non-adaptive (Diks & Pelc, 1996; Harutyunyan et al., 2011; Harutyunyan & Taslakian, 2004; J.-H. Kim & Chwa, 2005), or adaptive (Diks & Pelc, 1996; Rosenthal & Scheuermann, 1987).

(4) **Possibility of defining various fitness scores:** the proposed framework can accept any fitness score defined by the user. For instance, to minimize the broadcast time of a particular vertex

u under model M, the user can define $f_3(\sigma) = B_M^\sigma(u, G)$. Moreover, more complex fitness functions such as $f_4(\sigma) = f_1(\sigma) \times f_2(\sigma)$ could be defined according to the requirements of the network manager. This useful feature is not considered in other related works, such as (Hasson & Sipper, 2004; Hoelting et al., 1996; Lima et al., 2022; Sun & Li, 2006).

(5) **Efficiency:** the proposed framework is quite efficient in terms of run time. Our numerical results show that it could find optimal (or near-optimal) solutions for graphs with hundreds of nodes in a few seconds. Considering the limited resources used in this study, we believe this work is scalable to a reasonable extent for real-world applications.

(6) **Providing the broadcast scheme:** As opposed to several studies which propose closed formula upper bounds for some graphs alongside a descriptive broadcast scheme (Diks & Pelc, 1996; Harutyunyan et al., 2011; Harutyunyan & Taslakian, 2004; J.-H. Kim & Chwa, 2005; Rosenthal & Scheuermann, 1987), our method generates the actual broadcast scheme. This is quite useful for real-world scenarios. Also note that the generated schemes could be used to prove many theoretical aspects related to this problem. Moreover, the process of encoding/decoding of chromosomes does not require any knowledge on the topic, in contrast with other related works such as the one suggested in (Hoelting et al., 1996; Lima et al., 2022; Sun & Li, 2006).

8.4 Experimental Results

To show the efficiency of the proposed method, we have conducted several computer simulations. Table 8.5 shows the aims and scopes of each experiment.

8.4.1 Experiment 1

This experiment studies the impact of our algorithm's parameters on its performance. There are two major parameters in our algorithm:

- Population size $|p|$: As the size of the initial population increases, the chance of finding a better solution in early iterations gets higher. However, the computational cost associated

Table 8.5: Aims and scopes of the experiments

Experiment	What?	Why?	How?	Graph(s)
Experiment 1	Parameter Tuning	To see the impact of changing HUB-GA parameters on its performance.	For a graph G, we change parameters $\|p\|$ and S_t, while reporting $f_1(\sigma)$ and $f_2(\sigma)$ and the run time.	Karate club network (Zachary, 1977)
Experiment 2	Performance comparison vs. Classical model	To see whether the found broadcast time under universal lists model approaches its optimal value or not.	By calculating the ratio of $B_M(G)/B_{cl}(G)$ for different interconnection networks.	Well-known interconnection networks for which the value of $B_{cl}(G)$ is known.
Experiment 3	Performance comparison vs. degree-based heuristics	To see whether HUB-GA outperforms other degree-based heuristics or not.	By comparing the performance of our heuristic with three heuristics for clique-like structure graphs.	Clique-like graphs: Ring of cliques (Kamiński, Prałat, & Théberge, 2021), and Windmill graph (Bermond, 1979)
Experiment 4	Performance comparison vs. state-of-the-art heuristics	To see whether HUB-GA gets close to other heuristics for classical broadcasting or not.	By comparing the performance of our heuristic with two lower bounds and six upper bounds.	Interconnection Networks and Complex networks with small-world model (Rossi & Ahmed, 2015)

with the algorithm increases as well. Therefore, choosing the initial size of the population plays a vital role in the algorithm's performance.

- Stability S_t: The stopping criterion of the proposed algorithm is its stability over the last S_t iterations; that is, if the fitness score of the fittest individual does not change over S_t generations, the algorithm stops and returns the best solution. Choosing an appropriate number for S_t is a trade-off; the higher the value of S_t, the higher the chance of finding optimal solutions, and the longer it takes for the algorithm to terminate.

The Karate club network (Zachary, 1977) has been selected for this experiment with 34 nodes and 78 edges (See Figure 8.4). Then, we run our algorithm under all three models (na, a, fa) while changing those parameters. The parameter $|p|$ increases from 10 to 500, while we let S_t change from 1 to 15. The performance of the algorithm will be measured based on 4 criteria: The broadcast time (f_1), the run time required for calculating f_1, the average broadcast time (f_2), and the run time required for calculating f_2. The results of this experiment are reported in Figures 8.5 and 8.6.

Several exciting points could be drawn by analyzing the results given in Figure 8.5. Firstly, by

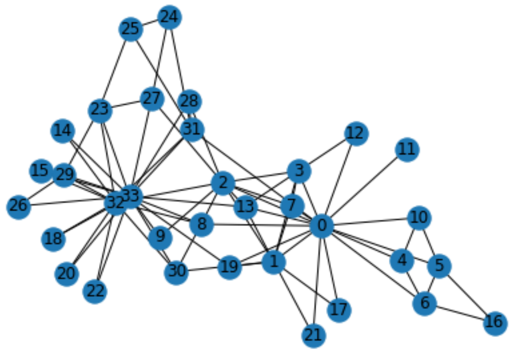
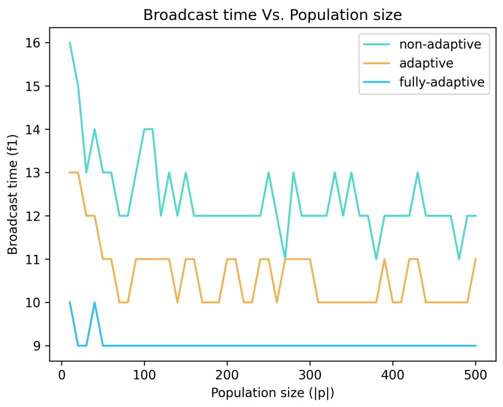
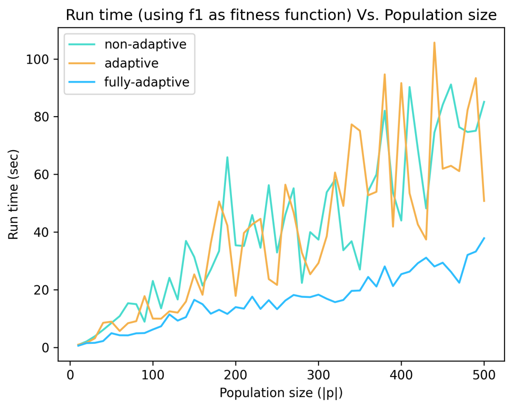


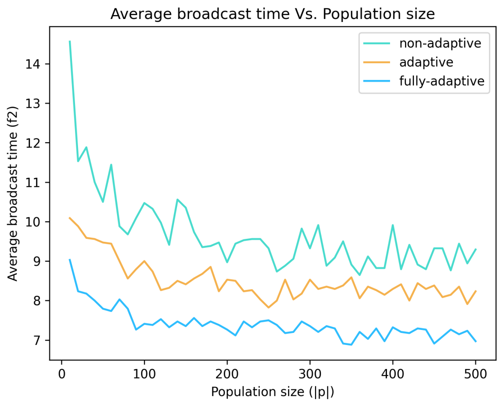
Figure 8.4: Karate club network (Zachary, 1977), used in Experiment 1



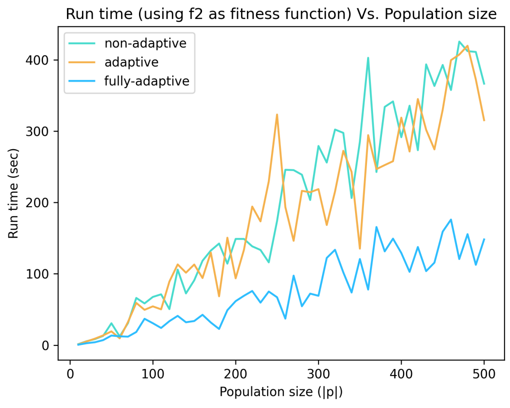
(a) Broadcast time (f_1)



(b) Run time for calculating f_1



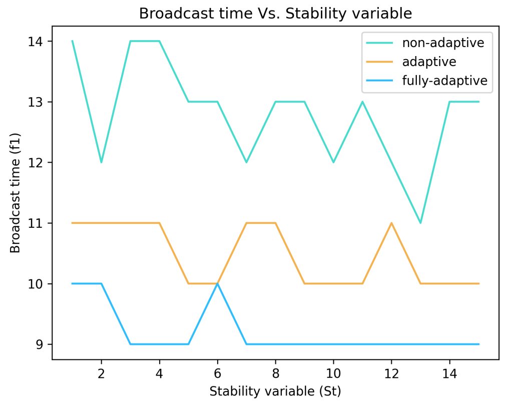
(c) Average broadcast time (f_2)



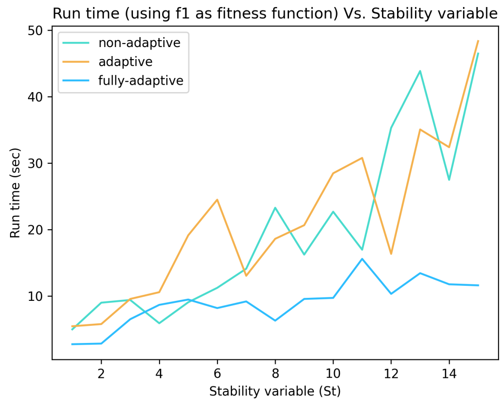
(d) Run time for calculating f_2

Figure 8.5: Results of Experiment 1: Impact of Population Size ($|p|$)

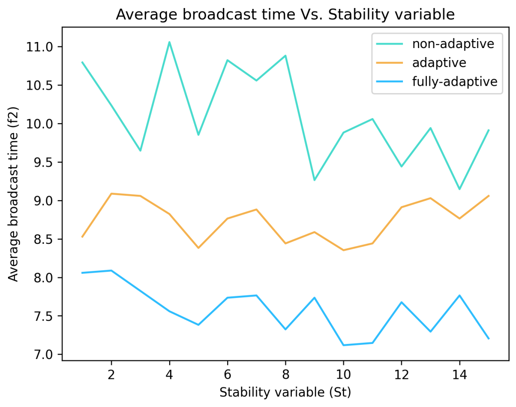
increasing the population size, the chance of minimizing fitness functions (either f_1 or f_2) improves. But the slope of this improvement lessens for higher values of $|p|$. In other words, increasing the



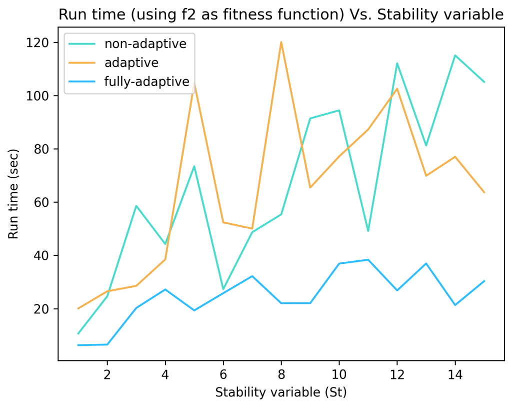
(a) Broadcast time (f_1)



(b) Run time for calculating f_1



(c) Average broadcast time (f_2)



(d) Run time for calculating f_2

Figure 8.6: Results of Experiment 1: Impact of Stability variable (S_t)

population size to a certain value, such as 200, could benefit the algorithm. Still, higher values of $|p|$ seem almost ineffective, while they need more processing time. For instance, we compare the performance of the fa model when $|p|$ increases from 200 to 500 considering f_2: although the run time increases by %140, the fitness score improves only by %4. These numbers are %434 and %3.5 under the adaptive and %146 and %3.6 under the non-adaptive model, respectively. Therefore, based on this experiment, we set the value of $|p|$ to 200 for the rest of this study.

Secondly, by comparing Figures 8.5 (a) and (c), it is clear that the average broadcast time (f_2) could be improved more, provided the fact that the algorithm has enough resources (time and computational power). Decreasing the broadcast time (f_1) looks pretty challenging, particularly once f_1 has reached a local minimum. This is due to the nature of the search space associated with f_1 and

f_2. For f_1, the search space has drastic steps, usually not very close to each other (since a very small change in σ might not affect $f_1(\sigma)$ sometimes). However, fitness function f_2 has a search space with many small steps since by a slight modification in σ, $f_2(\sigma)$ is expected to change slightly as well.

Thirdly, with a quick look at Figures 8.5 (b) and (d), it is concluded that the Fully-adaptive model is the quickest model among all. This idea has been proved before in Theorem 5.1.1. Broadcasting under this model is also affected less than two other models when changing $|p|$.

Also, we need to explain the reason for high fluctuations in Figures 8.5 (b) and (d): according to the randomness of `HUB-GA`, it is possible that during early generations, a fit chromosome is constructed and it will survive to future generations. Therefore, the final answer could be found quickly. On the other hand, if this chromosome is not generated in early iterations, it could take several generations to construct it from several good solutions. So, regardless of the population size, the run time of GA fluctuates. Though the range of this fluctuation widens as $|p|$ grows, according to Figures 8.5 (b) and (d).

The next parameter of our algorithm is S_t, whose effect on the performance of `HUB-GA` is reported in Figure 8.6. Similar to the population size, as S_t grows, the chance of finding a solution with a smaller broadcast time (f_1 or f_2) increases. However, increasing S_t to large values, such as 10 or higher, does not seem to have a notable effect on the performance of the algorithm while it requires more processing time. Consequently, based on the results of this experiment, we select a value of $S_t = 5$ throughout the rest of this study. Also, note that by looking at the line charts provided in Figure 8.6, the fully-adaptive model is the quickest.

8.4.2 Experiment 2

This experiment aims to compare the GA heuristic for the universal list model with known bounds on the classical model for commonly used interconnection networks. Therefore, we measure the ratio of $\frac{B_M^\sigma(G)}{B_{cl}(G)}$ for several graphs. Ultimately, the found broadcast time under universal list models should approach the desired value. For instance, consider line graph P_n on n vertices under the non-adaptive model. Recall that $B_{cl}(P_n) = n - 1$, and $B_{na}(P_n) = \lceil \frac{3n}{2} \rceil - 2$ (Diks & Pelc, 1996). If our algorithm finds the optimal scheme, as n approaches infinity, the ratio is supposed to approach 1.5: $\lim_{n\to\infty} \frac{B_{na}(P_n)}{B_{cl}(P_n)} = \lim_{n\to\infty} \frac{\lceil \frac{3n}{2} \rceil - 2}{n-1} = 1.5$. This ratio could be calculated easily for

Table 8.6: Results of Experiment 2

Graph G	size	$\frac{B^{\sigma}_{fa}(G)}{B_{cl}(G)}$	$\frac{B^{\sigma}_{a}(G)}{B_{cl}(G)}$	$\frac{B^{\sigma}_{na}(G)}{B_{cl}(G)}$
Path P_n	$2 \leq n \leq 1000$	**1.00***	**1.00***	**1.49***
Cycle C_n	$3 \leq n \leq 1000$	**1.00***	**1.00***	**1.32***
Star S_n	$2 \leq n \leq 1000$	**1.00***	**1.00***	**1.01***
Complete Graph K_n	$3 \leq n \leq 50$	1.14	1.39	1.42
Grid $G_{n\times m}$	$2 \leq n, m \leq 10$	1.07	1.08	1.35
Tori $T_{n\times m}$	$2 \leq n, m \leq 10$	1.09	1.24	1.55
Hypercube H_d	$2 \leq d \leq 9$	1.06	**1.41***	**1.68***
Cube Connected Cycle CCC_d	$2 \leq d \leq 7$	1.14	**1.18***	**1.52***
Shuffle Exchange SE_d	$3 \leq d \leq 9$	**1.06***	**1.09***	**1.44***
De Bruijn DB_d	$3 \leq d \leq 9$	**1.09***	**1.18***	**1.51***

other graphs based on the results reported in Table 7.2.

To this aim, for a graph G on n vertices (or dimension d, where n is a function of dimension d), we run our heuristic under all three models (na, a, fa), then compare the achieved value with the known value of $B_{cl}(G)$. We repeat this process for several instances of that graph and report the average of the ratio as the result. Note that for this experiment, we need to use some graphs for which the exact value of classical broadcast time is known; thus, the ratio could be calculated. The only exception is the De Bruijn graph DB_d for which the exact value of $B_{cl}(G)$ is unknown. Hence, we compare our results with the best lower bound suggested in (Klasing et al., 1994). The results of this experiment are reported in Table 8.6. An asterisk means that our algorithm has been able to find the optimal scheme or a scheme better than the current upper bound for a given graph.

Table 8.6 shows that HUB-GA is able to achieve the optimal result for Path, Star, and Cycle under all three models since the ratio approaches the optimal value. For Grid and Tori, however, the values are close to optimal. The reason is that an optimal broadcast scheme for these networks is symmetric for all nodes. Finding a completely symmetric scheme for a large graph could be difficult for the GA due to its randomness. Also, in four well-known interconnection networks, H_d, CCC_d, DB_d, and SE_d, our algorithm outperforms the best upper bound reported in Table 7.2, except for the fully-adaptive model for H_d and CCC_d, for which we proved the exact values of broadcast time earlier. Besides, the achieved broadcast time for all these networks is mostly between 1 to 1.5 times its classical time, which is very promising. A trivial upper bound for the non-adaptive broadcast time of an arbitrary graph is suggested in Theorem 2.2. of (Diks & Pelc, 1996) as follows: $B_{na}(G) \leq 3 \cdot B_{cl}(G)$, for any graph G.

The results of the complete graphs are discussed hereafter. Firstly, note that the known bound for a complete graph is as follows:

$$\lceil \log n \rceil = B_{cl}(K_n) \leq B_{fa}(K_n) \leq B_a(K_n) \leq B_{na}(K_n) \leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil \tag{34}$$

Where the lower bound comes from the trivial lower bound on broadcast time, and the upper bound is presented in (Diks & Pelc, 1996). According to Table 8.6, the result of each model differs from other models. Hence, in equation (34), the inequalities are more likely to become strictly less than, particularly for the fully-adaptive model. Based on this result, we make the following conjecture:

Conjecture 8.4.1. *For a sufficiently large n, the broadcast time of a complete graph K_n is bounded as follows:*

$$\lceil \log n \rceil = B_{cl}(K_n) < B_{fa}(K_n) < B_a(K_n) \leq B_{na}(K_n) \leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil \tag{35}$$

To experimentally validate this conjecture, we designed another experiment in which the performance of our GA is compared with the lower and upper bounds of Equation (34) for a complete graph K_n on size $3 \leq n \leq 150$. The results of this experiment are illustrated in Figure 8.7. First, observe that these results correspond to the bounds of Equation (34). Secondly, based on this result, the smallest value of n for which $\lceil \log n \rceil = B_{cl}(K_n) < B_{fa}(K_n)$ is $n = 8$ where $B^{\sigma}_{fa}(K_8) = 4$, while $B_{cl}(K_8) = 3$. To prove that $B_{fa}(K_8) > 3$ (which also proves the first inequality of Equation (35)), one solution is to examine all possible broadcast schemes for a complete graph K_8 under the fully-adaptive model. If no scheme σ could be found that yields $B^{\sigma}_{fa}(K_8) = 3$, the proof is completed. However, by considering Equation (30), there are almost $2.08e40$ unique broadcast schemes for K_8, which is impossible to be generated without a careful constructive method. Thirdly, the truth of the second inequality could be realized by observing the growth rate of $B_{fa}(K_n)$ and comparing it with that of $B_a(K_n)$ in Figure 8.7.

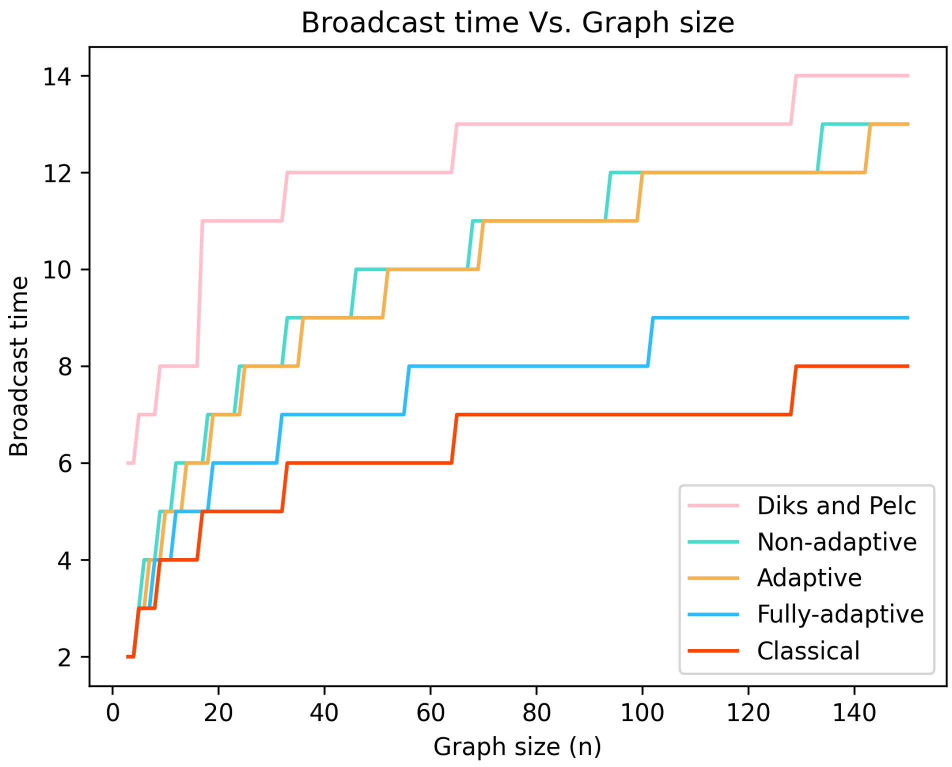


Figure 8.7: Comparing the performance of HUB-GA with the known bounds on the broadcast time of a complete graph K_n of size $3 \leq n \leq 150$.

We need to stress the importance of Conjecture 8.4.1: if it is true, it implies that the definition of broadcast graphs (*bg*'s) are different under the universal list models. A *bg* on n vertices is a graph for which broadcasting could be finished in $\lceil \log n \rceil$ time units starting from any originator. In the classical model, the most trivial *bg* on n vertices is K_n in which $\forall u \in V(K_n) : B_{cl}(u, K_n) = \lceil \log n \rceil$. If Conjecture 8.4.1 is true, it means that: 1) complete graphs are not immediate bg's under the universal lists model, 2) for an arbitrary value of n, the existence of a graph on n vertices for which the $\lceil \log n \rceil$ time could be obtained under the universal list model is questionable. We studied this problem in Chapter 7, where the Hypercube H_d was introduced as the first infinite family of mbg's under the fully-adaptive model for any $n = 2^d$. We also proposed different bg's for the following values of n: $n = 2^{k-1} + 2^{k-2}$, $n = 2^{k-1} + 2^{k-3}$, $n = 2^{k-1} + 2^{k-4}$, and $n = 2^{k-1} + 2^{k-2} + 2^{k-3}$ for any integer $k = \lceil \log n \rceil \geq 4$. However, the puzzling question discussed in this study concerning arbitrary values of n still remains open.

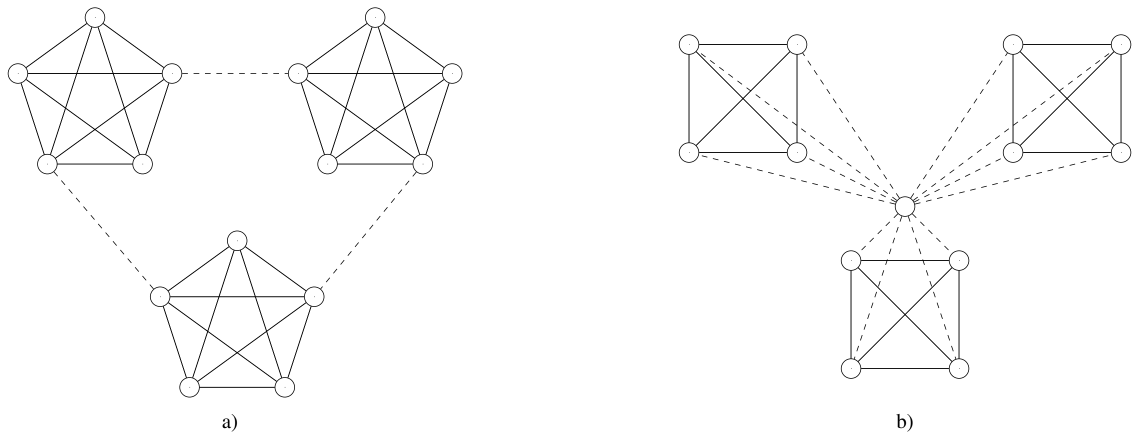


Figure 8.8: a) Ring of Clique $RC_{3,5}$, b) Windmill graph $W_{5,3}$

8.4.3 Experiment 3

Observe that for a fixed value of n, the function discussed in Equation (30), i.e., the size of the search space of this problem, reaches its maximum when the degree of each node maximizes, or a complete graph on n vertices. In fact, as opposed to the classical model in which $B_{cl}(K_n)$ is known, there is no optimal bound for these networks under universal list models. Besides, as mentioned in (Harutyunyan & Maraachlian, 2008), the broadcast problem becomes more complicated when there are several intersecting cycles. In the previous experiment, we studied this problem for complete graphs K_n. Subsequently, this experiment aims to analyze the performance of `HUB-GA` for graphs with clique-like subgraphs.

To this aim, two different graphs are considered: the Ring of cliques (Kamiński et al., 2021) and the Windmill graph (Bermond, 1979). A Ring of Clique $RC_{n,m}$ consists of n cliques of size m that are connected to each other on a cycle. It has $n \cdot m$ vertices and $n \cdot \frac{m \cdot (m-1)}{2} + n$ edges. Also, a Windmill graph $W_{k,n}$ is a graph of n cliques each of size k that are all joined at one vertex. Alternatively, one may generate n cliques of size $k-1$ and add one node to this graph which is connected to all other vertices. A $W_{k,n}$ has $n \cdot (k-1) + 1$ vertices and $\frac{nk(k-1)}{2}$ edges. Figure 8.8 portrays $RC_{3,5}$ and $W_{5,3}$.

The classical broadcast time of these networks is not known. Therefore, we cannot make a similar experiment as the previous experiment. Instead, we compare the performance of `HUB-GA` with three heuristics:

- **`Ran. Seq.`:** The ordering of a vertex is uniformly random.
- **`Inc. Deg.`:** Neighbors of a vertex are sorted in ascending order based on their degree.
- **`Dec. Deg.`:** Neighbors of a vertex are sorted in descending order based on their degree.

The idea of degree-based heuristics is claimed to be efficient under the classical model, such as the `AM` algorithm provided in (Scheuermann & Wu, 1984). Besides, not only the worst behavior of random broadcasting in a network, a.k.a messy broadcasting, has received a lot of attention from researchers (Ahlswede et al., 1994; Comellas et al., 2003; Harutyunyan & Liestman, 1998), but also the average-case random broadcasting time of various networks are studied in (C. Li et al., 2008).

For each heuristic, we report three numbers: **Min** which is the minimum broadcast time of any vertex in the network using the scheme provided by a heuristic: $\min_{u\in V(G)}\{B_M^\sigma(u,G)\}$, **Avg** or the average broadcast time of all vertices of the graph following that scheme: $\frac{\sum_{u\in V(G)}\{B_M^\sigma(u,G)\}}{n}$, and **Max** which is the broadcast time of the worst originator in the graph following a particular scheme: $\max_{u\in V(G)}\{B_M^\sigma(u,G)\}$. Note that **Max** corresponds to fitness function f_1, whereas **Avg** corresponds to f_2. The results of this experiment are reported in Table 8.7 for $RC_{n,m}$ and Table 8.8 for $W_{k,n}$, where $3 \leq n, m, k \leq 6$. As the performance of the `Ran. Seq.` is random in each run; we average its result over five runs.

By looking at the results expressed in Table 8.7, `HUB-GA` outperforms all other heuristics in terms of **Min**, **Avg**, and **Max** broadcast time in almost all cases. Moreover, among three competitor heuristics, `Dec. Deg.` is the most successful one for this particular graph family, while `Inc. Deg.` is even worse than pure random ordering. The reason is that by using `Dec. Deg.` a vertex will send the message to join vertices first since they have the highest degree in the graph (Figure 8.8-a, there are 6 join vertices). Then, it will send the message to its other neighbors randomly. This is an efficient ordering, which could be optimal with a careful ordering of other neighbors. On the other hand, the `Inc. Deg.` heuristic is the least successful one since a vertex prioritizes its non-joint neighbors, which is not optimal behavior.

Lastly, the margin between the performance of the GA and `Dec. Deg.` heuristic is wider under the non-adaptive model compared to the fully-adaptive model. The underlying reason is interesting: consider a random scheme for non-adaptive broadcasting in a complete graph, and

Table 8.7: Results of Experiment 3 for $RC_{n,m}$, where $3 \leq n, m \leq 6$

$RC_{n,m}$		$\lvert V\rvert$	$\lvert E\rvert$	Non-adaptive model											
				Min				Avg				Max			
n	m			`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`
3	3	9	12	**5**	**5**	6	**5**	6.22	6.11	6.77	**5.33**	7.4	7	7	**6**
4	3	12	16	6.6	7	9	**6**	8.38	7.91	9.41	**6.83**	10.4	9	10	**8**
5	3	15	20	8.4	8	10	**7**	9.93	9	10.86	**8.33**	12	**10**	12	**10**
6	3	18	24	10.2	**9**	12	**9**	12.04	10.44	13.77	**9.66**	13.8	12	15	**10**
3	4	12	21	6.2	6	9	**5**	8.06	7.08	9.5	**5.83**	9.6	8	10	**7**
4	4	16	28	8.4	**7**	12	**7**	10.33	8.81	13.56	**7.81**	12.6	10	15	**9**
5	4	20	35	10.8	**8**	15	**8**	13.05	10.2	16.1	**9.45**	15.8	**12**	17	**12**
6	4	24	42	12.2	**10**	18	**10**	14.91	11.79	19.95	**11.29**	17.8	14	22	**13**
3	5	15	33	7.2	7	12	**6**	9.78	7.93	12.4	**7.2**	11.8	**9**	14	**9**
4	5	20	44	8.8	8	16	**7**	12.37	9.9	17.75	**8.65**	15.4	**11**	19	**11**
5	5	25	55	11.6	9	21	**8**	14.55	11.12	21.84	**10.04**	17.6	13	23	**12**
6	5	30	66	14	**11**	24	**11**	17.91	12.73	26.86	**12.13**	21.8	**14**	29	**14**
3	6	18	48	8.8	8	15	**6**	10.94	8.77	15.77	**7.55**	13	10	17	**9**
4	6	24	64	11	9	20	**7**	14.52	10.29	21.83	**9.2**	17.8	12	24	**11**
5	6	30	80	14	11	25	**9**	17.22	12	26.53	**10.8**	20	**13**	27	**13**
6	6	36	96	16.4	**11**	30	**11**	20.59	13.66	33.05	**12.44**	23.8	16	36	**14**

$RC_{n,m}$		$\lvert V\rvert$	$\lvert E\rvert$	Adaptive model											
				Min				Avg				Max			
n	m			`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`
3	3	9	12	4.4	**4**	5	**4**	4.8	**4.33**	5	**4.33**	5	**5**	5	**5**
4	3	12	16	5.6	**5**	6	**5**	6.4	**5.33**	6.66	**5.33**	7	**6**	7	**6**
5	3	15	20	6.8	**6**	8	**6**	7.58	**6.33**	8	**6.33**	8	**7**	8	**7**
6	3	18	24	8	**7**	9	**7**	8.86	**7.33**	9.38	7.61	10	**8**	10	**8**
3	4	12	21	5.6	5	6	**4**	6.13	5.41	6.83	**5.08**	6.8	**6**	7	**6**
4	4	16	28	7	6	9	**5**	8.36	6.5	9.12	**6.25**	9.4	**7**	10	**7**
5	4	20	35	8.4	7	11	**6**	9.77	**7.25**	11	**7.25**	10.8	**8**	11	**8**
6	4	24	42	10.4	**8**	12	**8**	12.15	**8.29**	12.91	8.41	13.6	**9**	13	**9**
3	5	15	33	6.6	**5**	8	**5**	7.71	6	8.13	**5.6**	8.6	7	9	**6**
4	5	20	44	7.8	**6**	11	**6**	9.45	7.35	11.35	**6.8**	11	**8**	12	**8**
5	5	25	55	9.8	**7**	12	**7**	11.52	**7.64**	12.4	7.96	13.2	**8**	13	9
6	5	30	66	12	**8**	14	**8**	13.97	9.16	15.26	**9.03**	15.8	**10**	17	**10**
3	6	18	48	6.8	**6**	9	**6**	8.46	7.05	9.55	**6.44**	9.6	8	10	**7**
4	6	24	64	9	7	13	**6**	11.12	7.69	13.04	**7.45**	13	**8**	14	**8**
5	6	30	80	11	**8**	14	**8**	13.15	8.93	14.5	**8.86**	14.8	**10**	15	**10**
6	6	36	96	13.8	9	17	**8**	15.66	9.83	18	**9.8**	17.6	**11**	19	**11**

$RC_{n,m}$		$\lvert V\rvert$	$\lvert E\rvert$	Fully-adaptive model											
				Min				Avg				Max			
n	m			`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`
3	3	9	12	4.4	**4**	5	**4**	4.86	**4.33**	5	**4.33**	5	**5**	5	**5**
4	3	12	16	5.8	**5**	6	**5**	6.25	**5.33**	6.41	**5.33**	7	**6**	7	**6**
5	3	15	20	6.6	**6**	8	**6**	7.53	**6.33**	8	**6.33**	8	**7**	8	**7**
6	3	18	24	8	**7**	9	**7**	8.73	**7.33**	9.44	**7.33**	9.2	**8**	10	**8**
3	4	12	21	5.4	**4**	6	5	5.9	**5**	6.66	**5**	6.6	6	7	**5**
4	4	16	28	6.6	**5**	8	**5**	7.82	5.87	8.81	**5.5**	9	7	10	**6**
5	4	20	35	8.2	**6**	11	**6**	9.36	7.1	11	**7.05**	10.4	**8**	11	**8**
6	4	24	42	10.2	**7**	13	**7**	11.23	8.25	13.29	**8.16**	13	**9**	14	**9**
3	5	15	33	6	**5**	7	**5**	6.7	**5.6**	7.86	5.73	7.2	**6**	9	**6**
4	5	20	44	7.6	**6**	9	**6**	9.05	6.66	9.9	**6.65**	10.2	**7**	11	**7**
5	5	25	55	9.4	**7**	12	**7**	10.7	7.92	13.04	**7.8**	12.2	**9**	14	**9**
6	5	30	66	11	**8**	14	**8**	12.54	**8.6**	15.03	9.16	14	**9**	16	10
3	6	18	48	6.8	6	8	**5**	7.76	6.66	9.27	**5.88**	8.6	**7**	10	**7**
4	6	24	64	8.6	**7**	12	**7**	10.17	7.95	12.83	**7.5**	11.6	9	14	**8**
5	6	30	80	10.4	**7**	13	8	11.66	**8.26**	14.03	9.1	13	**9**	15	10
6	6	36	96	11.8	**8**	16	9	13.71	**9.25**	17.16	10.27	15.4	**10**	19	11

denote the non-adaptive broadcast time achieved by this scheme by t_{na}^{ran}. Also, denote the optimal broadcast time for the same graph under the same model by t_{na}^{opt}. Moreover, denote by t_{fa}^{ran} the fully-adaptive broadcast time of a random scheme for the same graph, whereas t_{fa}^{opt} is the optimal fully-adaptive broadcast time. We argue that the value of $t_{fa}^{ran} - t_{fa}^{opt}$ is more likely to be less than $t_{na}^{ran} - t_{na}^{opt}$ with infinite number of same experiments. This is because, in the fully-adaptive model, an inefficient behavior is to hit the same receiver by several senders. However, in the non-adaptive model, the message is likely to be sent back and forth between senders and receivers, resulting in a wider gap between the optimal scheme and a randomly selected ordering. Based on this observation, we expect the results of `HUB-GA` to have a wider margin with all three heuristics under the non-adaptive model compared to the fully-adaptive model in which the performance of all four heuristics could get much closer. The numerical results of this experiment reported in Table 8.7 also correspond to this observation.

Table 8.8 gives the results of the same experiment for Windmill graph $W_{k,n}$, where $3 \leq k, n \leq 6$. In $W_{k,n}$, there are two types of vertices: Join vertex (with a degree of $k(n-1)$), and other vertices (with a degree of $n-1$). Therefore, in competitor heuristics, the join vertex will choose a random ordering for its neighbors since all neighbors have the same degree. However, other vertices behave differently in two degree-based heuristics: in the `Inc. Deg.` heuristic, they will first distribute the message within their clique and then will send it to the join vertex, whereas this ordering is reversed in the `Dec. Deg.` heuristic. Therefore, we expect `Dec. Deg.` to perform slightly better than `Inc. Deg.`

The results in Table 8.8 correspond to our expectations, in which `Dec. Deg.` outperforms `Inc. Deg.` and `Ran. Seq.` heuristics under all three models. Interestingly, a random ordering of the vertices seems more effective than the `Inc. Deg.` heuristic. This is due to the missing call toward the join vertex during early stages under the increasing degree heuristic. All in all, the performance of `HUB-GA` is better than all three heuristics, particularly under the non-adaptive model, for the same reason discussed earlier.

Also, Figures 8.9 and 8.10 compare the results of `HUB-GA` with three heuristics for these two families of graphs. The comparison metric is the performance improvement which is calculated as follows for model M when comparing the performance of `HUB-GA` against heuristic `H`:

Table 8.8: Results of Experiment 3 for $W_{k,n}$, where $3 \leq k, n \leq 6$

$W_{k,n}$		$\lvert V\rvert$	$\lvert E\rvert$	Non-adaptive model											
				Min				Avg				Max			
k	n			`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`
3	3	7	9	4.4	5	**4**	**4**	5.94	5.57	5.42	**4.71**	7	**6**	**6**	**6**
4	3	9	12	5.8	7	8	**5**	7.2	8.44	9.33	**5.88**	7.8	9	10	**7**
5	3	11	15	7.6	9	8	**6**	8.96	9.72	9.63	**6.81**	10	10	10	**8**
6	3	13	18	8.8	10	10	**7**	11.49	11.61	11.53	**7.92**	12.4	12	12	**9**
3	4	10	18	5.6	6	7	**4**	7.72	7.3	8.8	**5.1**	9.2	8	10	**6**
4	4	13	24	6.2	6	6	**5**	9.12	7.46	8.53	**6.07**	11	8	9	**7**
5	4	16	30	8	8	9	**6**	11.23	10.37	11.25	**7.31**	13	11	12	**8**
6	4	19	36	11.6	10	11	**8**	14.53	14.15	15.89	**8.84**	16.2	15	17	**10**
3	5	13	30	6.4	7	9	**5**	9.33	9.69	11.46	**6.07**	11.4	11	13	**8**
4	5	17	40	6.8	7	9	**6**	9.95	10	11.82	**7.11**	12.4	11	13	**9**
5	5	21	50	10.4	9	8	**7**	14.45	10.57	11.61	**8.14**	16.8	11	12	**9**
6	5	25	60	11	11	11	**8**	14.4	13.48	14.52	**9.08**	16.6	14	15	**11**
3	6	16	45	6.6	8	7	**5**	9.51	8.62	11.37	**6.37**	11.8	9	12	**8**
4	6	21	60	9.4	10	9	**6**	12.49	10.95	13.76	**7.33**	14.6	11	14	**9**
5	6	26	75	11.6	12	12	**7**	15.6	14.38	16.42	**8.53**	18.2	15	17	**10**
6	6	31	90	13.2	16	18	**9**	17.54	21	22.87	**9.93**	19.6	22	24	**11**

$W_{k,n}$		$\lvert V\rvert$	$\lvert E\rvert$	Adaptive model											
				Min				Avg				Max			
k	n			`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`
3	3	7	9	4.2	**4**	**4**	**4**	4.82	**4.28**	4.85	**4.28**	5.4	**5**	**5**	**5**
4	3	9	12	5.8	6	7	**5**	6.35	6.77	7.44	**5.22**	7.2	7	8	**6**
5	3	11	15	6.6	**6**	7	**6**	7.92	7	7.72	**6.18**	9	8	8	**7**
6	3	13	18	8	8	9	**7**	8.7	8.69	9.46	**7.15**	9.4	10	10	**8**
3	4	10	18	5.2	5	7	**4**	6.65	6.2	7.6	**4.7**	7.8	7	8	**5**
4	4	13	24	6.6	6	7	**5**	7.5	6.92	8.38	**5.38**	8.6	8	9	**6**
5	4	16	30	8	7	9	**6**	9.62	9.5	11.12	**7**	11	11	12	**8**
6	4	19	36	9.2	**7**	8	**7**	10.41	7.78	9.89	**7.21**	11.6	**8**	10	**8**
3	5	13	30	6.2	6	8	**5**	7.53	7.23	9.3	**5.69**	8.6	8	10	**7**
4	5	17	40	6.8	8	10	**6**	8.71	9.76	12.17	**6.47**	10.2	11	14	**7**
5	5	21	50	8.6	8	9	**7**	10.11	9	11.33	**7.66**	11.6	10	13	**9**
6	5	25	60	10.6	10	11	**8**	12.31	11.2	13.2	**8.4**	14.2	12	15	**9**
3	6	16	45	6.2	5	6	**5**	8.01	7	8.68	**5.68**	9.6	8	10	**6**
4	6	21	60	8.6	8	10	**6**	10.72	9.19	12	**6.9**	12.2	10	13	**8**
5	6	26	75	9.2	9	11	**7**	11.36	10.84	13.76	**7.88**	13	12	15	**9**
6	6	31	90	10.6	12	13	**8**	12.58	12.93	15.16	**8.61**	14.4	14	17	**10**

$W_{k,n}$		$\lvert V\rvert$	$\lvert E\rvert$	Fully-adaptive model											
				Min				Avg				Max			
k	n			`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`	`Ran.Seq.`	`Dec.Deg.`	`Inc.Deg.`	`HUB-GA`
3	3	7	9	4.2	**4**	**4**	**4**	4.91	**4.14**	4.85	**4.14**	5.6	**5**	**5**	**5**
4	3	9	12	5.4	**5**	6	**5**	6.06	6.11	6.88	**5.11**	7	7	8	**6**
5	3	11	15	6.4	**6**	6	**6**	6.96	**6.09**	6.9	**6.09**	7.6	**7**	**7**	**7**
6	3	13	18	8	8	9	**7**	8.9	8.766	9.61	**7.07**	9.8	10	10	**8**
3	4	10	18	5.2	5	6	**4**	6.21	5.6	6.9	**4.4**	7.2	7	8	**5**
4	4	13	24	6	6	7	**5**	7.3	6.92	8.38	**5.23**	8.6	8	9	**6**
5	4	16	30	6.4	**6**	8	**6**	8	7.68	9.31	**6.18**	9.4	9	10	**7**
6	4	19	36	8.2	8	9	**7**	9.69	8.78	10.42	**7.21**	11	10	11	**8**
3	5	13	30	5.6	**5**	6	**5**	6.89	6.15	7.92	**5.3**	8.2	7	9	**6**
4	5	17	40	7.2	**6**	7	**6**	8.43	7.17	9.05	**6.23**	9.8	8	10	**7**
5	5	21	50	7.4	**7**	8	**7**	8.91	8.04	10.14	**7.28**	10.2	9	11	**8**
6	5	25	60	9	**8**	9	**8**	10.87	9.16	11.4	**8.12**	12.6	10	12	**9**
3	6	16	45	6.2	7	7	**5**	7.45	7.18	9.5	**5.56**	8.6	8	10	**7**
4	6	21	60	7.6	7	8	**6**	9.53	8.28	10.76	**6.52**	11	9	12	**7**
5	6	26	75	9.8	11	13	**7**	11.51	12.15	14.73	**7.34**	13.4	14	16	**8**
6	6	31	90	10	11	12	**8**	12.19	11.67	14.35	**8.58**	13.6	13	16	**10**

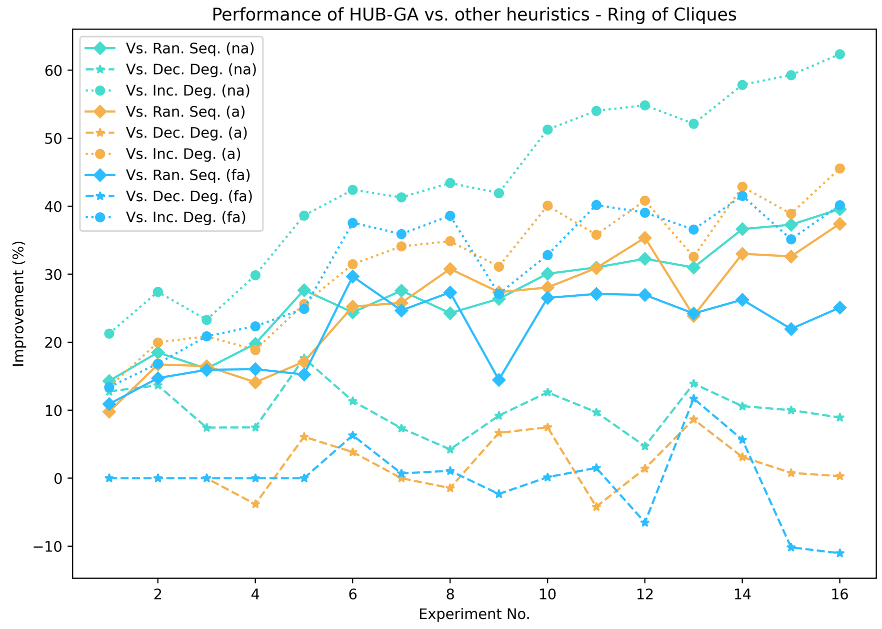


Figure 8.9: Comparison of the performance of the HUB−GA and other heuristics in terms of Average broadcast time - Ring of cliques

$$\text{improvement}_M^{\text{H}} = \frac{\textbf{Avg}(B_M^{\text{H}}(G)) - \textbf{Avg}(B_M^{\texttt{HUB-GA}}(G))}{\textbf{Avg}(B_M^{\texttt{HUB-GA}}(G))} \tag{36}$$

In which $M \in \{na, a, fa\}$ and H$\in$ $\{$`Ran.Seq., Dec.Deg., Inc.Deg.`$\}$.

As can be seen, HUB−GA is able to speed up the broadcast process up to almost %60 compared to degree-based heuristics for both networks. The main reason for high fluctuations in Fig. 8.9 and 8.10 are as follows:

(1) **Randomness of HUB−GA:** As mentioned earlier, GA is a random search algorithm that starts with generating several random solutions for the problem at its first step. Although GA is expected to approach a nearly-optimal solution in the long run, it is possible that it does not find it in some situations. Therefore, the performance of HUB−GA is random. However, our extensive experiments show that our heuristic is quite effective in dealing with real-world data sets.

(2) **Randomness of other heuristics:** When the performance of HUB−GA is compared with

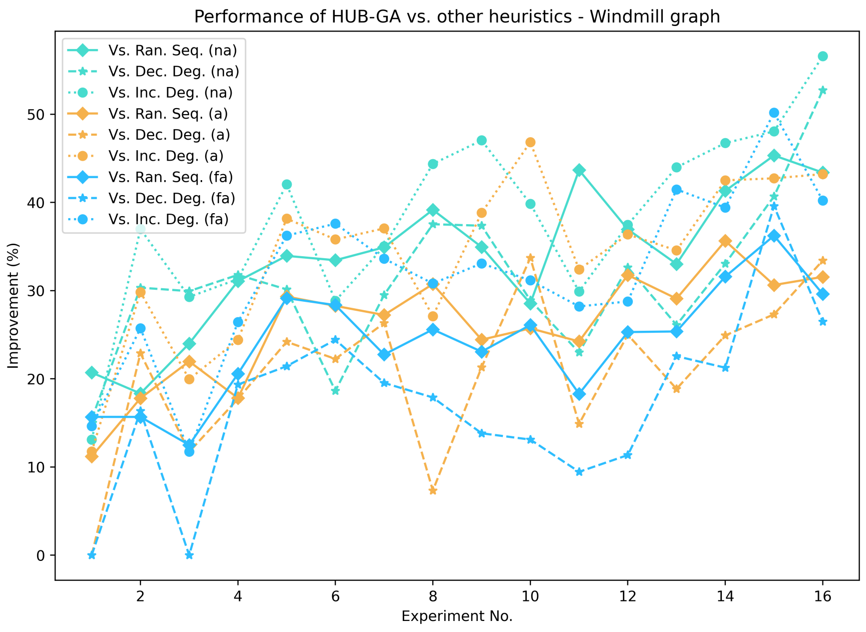


Figure 8.10: Comparison of the performance of the `HUB-GA` and other heuristics in terms of Average broadcast time - Windmill graph

`Ran.Seq.` the fluctuations are wider. This is due to the pure randomness of `Ran.Seq.` heuristic. Even though we took the average results of `Ran.Seq.` over 5 runs, it still exhibits its random nature which leads to high fluctuations.

(3) **Random ordering of deterministic heuristics:** Consider other deterministic heuristics, `Dec.Deg.` or `Inc.Deg.`. In both heuristics, the ties are broken randomly. In graphs with several vertices with the same degree (such as the Windmill graph), these heuristics are almost random, with the only difference being the position of the joint vertex. This fact justifies more drastic fluctuations in Fig. 8.10 (Windmill graph) compared to Fig. 8.9 (Ring of cliques).

(4) **x-axis:** In Fig. 8.9 and 8.10, the x-axis is the experiment number. According to Table 8.7 and 8.8, the experiment number does not have a monotonic relation with either $|V|$ or $|E|$. However, we decided to choose the experiment number as the x-axis (and for instance, not sort the experiments based on $|E|$) to avoid confusion. We believe this is the best way to represent our findings.

8.4.4 Experiment 4

This experiment compares our heuristic with some state-of-the-art heuristics in the literature. Since the instances used in previous works, such as (Hasson & Sipper, 2004; Hoelting et al., 1996; Scheuermann & Wu, 1984) are no longer available, we used two types of networks:

(1) **Interconnection Networks (44 instances):** We replicated several instances from (de Sousa et al., 2018; Harutyunyan & Jimborean, 2014; Harutyunyan & Wang, 2010; Hasson & Sipper, 2004; Lima et al., 2022). These networks include 8 instances of Hypercube H_d, 5 instances of Cube Connected Cycle CCC_d, 7 instances of De Bruijn DB_d, 8 instances of Shuffle Exchange SE_d [1], and 16 instances of Harary graph $H_{k,n}$ [2].

(2) **Connected Complex Networks (30 instances):** well-established instances of connected complex networks from Network Repository[3] (Rossi & Ahmed, 2015). We consider various instances based on small-world networks with 100 nodes to address various industry scenarios (Freitas, Aquino, Ramos, Frery, & Rosso, 2019; Lima et al., 2022). Note that the small-world model could be used to represent communication networks in real-world applications, as suggested by (Cabral, Aquino, Frery, Rosso, & Ramírez, 2013; Guidoni, Mini, & Loureiro, 2010; Lima et al., 2022).

For each instance we report $|V|$ (the number of vertices), $|E|$ (the number of edges), and edge density of the graph $\frac{2|E|}{|V|(|V|-1)}$. As mentioned before, our heuristic is the first of its kind in the literature for broadcasting with the universal lists model. Thus, we can only compare our heuristic with similar results under the classical model. We compared our results with two lower bounds:

- **`TLB`:** The trivial lower bound on the classical broadcast time of any graph G on n vertices: $B_{cl}(G) \geq \lceil \log n \rceil$.
- **`LBB` (Lima et al., 2022):** The lower bound on the classical model suggested in (Lima et al., 2022), namely `LBB-BFS`. The goal here is to find the maximum shortest path between any

[1]Data available here: https://github.com/sabergholami/
[2]Data available here: https://github.com/alfredolimams/
[3]Data available here: https://networkrepository.com/rand.php/

receiver and the originator (vertex v) by performing a BFS algorithm. Then, a lower bound on the value of $B_{cl}(v, G)$ is found, which is also a valid lower bound on the $B_{cl}(G)$.

Also, we compare our results with six upper bounds reported in the literature:

- **`TreeBlock` (de Sousa et al., 2018, 2020):** the constructive heuristic proposed in (de Sousa et al., 2018), with a more detailed version presented in (de Sousa et al., 2020) considering various network families. `TreeBlock` works based on detecting cut-points of graph G and breaking the problem down into several smaller problems in trees that are formed from the cut-points.

- **`NTBA` (Harutyunyan & Wang, 2010):** This algorithm works based on forming a layer graph from the original graph by performing a BFS. Then, it traverses the layer graph and produces a broadcast scheme for the originator.

- **`NEWH` (Harutyunyan & Jimborean, 2014):** This algorithm builds upon the idea of `NTBA` but applies a new non-random strategy to generate a spanning tree for broadcasting. The advantage of this algorithm is that almost half of vertices are informed by the shortest path from the originator, while the rest are informed by a path at most 3 hops longer.

- **`ILP` (de Sousa et al., 2018; Lima et al., 2022):** Consider exact algorithms on the classical broadcast problem. In addition to the dynamic programming algorithm presented in (Scheuermann & Wu, 1984), the most successful approach is the Linear Programming model of de Sousa et al. (de Sousa et al., 2018). The `ILP` algorithm (Lima et al., 2022) works based on the algorithm suggested in (de Sousa et al., 2018), with the difference of having possibly more than one originator. It is an Integer Linear Programming algorithm for the classical broadcast problem, solved by IBM Cplex 12.9.

- **`ACS` (Hasson & Sipper, 2004):** The Ant Colony System algorithm proposed by Hasson and Sipper (Hasson & Sipper, 2004). Note that the results reported by the authors of (Hasson & Sipper, 2004) illustrate that `ACS` outperforms the algorithm by Hoelting et al. (Hoelting et al., 1996). Thus, we did not consider their algorithm.

- **BRKGA (Lima et al., 2022):** The Biased Random Key Genetic Algorithm proposed in (Lima et al., 2022) with the First Receive First Send decoder. Considering the results reported in the corresponding study, this could be considered the most successful and recent heuristic for the classical broadcast problem. BRKGA_FRFS is believed to outperform several well-known heuristics for the classical broadcast problem such as Tree Block (de Sousa et al., 2018, 2020), NTBA (Harutyunyan & Wang, 2010), and NEWH (Harutyunyan & Jimborean, 2014).

The numerical results of this experiment are reported in Tables 8.9 and 8.10. The results of all lower and upper bounds are reproduced from their original paper, respectively. A hyphen in Tables 8.9 and 8.10 indicates that we do not have the result for an instance or the method could not produce a feasible solution. Also, if the result of a particular algorithm is not reported in the original paper for a sample, we used the result reported in (Lima et al., 2022), as they re-implemented all algorithms.

We need to point out that the value achieved by all upper bounds is for $B_{cl}(v, G)$ for a particular v as the originator (for most instances, $v = \{1\}$ as the originator). Recall that $\forall v : B_{cl}(v, G) \leq B_{cl}(G)$. Also, from Equation (28), it is clear that the value of $B^{\sigma}_{M}(G)$ is lower bounded by $B_{cl}(G)$ for $M = \{fa, a, na\}$. Thus, the values achieved by HUB-GA are not supposed to outperform the six upper bounds. In fact, it could be impossible to achieve those values in some cases in which the originator is not the worst originator. Finally, in all competitor heuristics, the goal is to minimize the broadcast time of a vertex in a given graph. This somewhat contrasts with the nature of the universal list model, in which the goal is to optimize the behavior of all nodes simultaneously with one universal list. However, we performed this experiment to see whether $B^{\sigma}_{M}(G)$ could get close to $B_{cl}(v, G)$ or not.

Our numerical results demonstrate that as opposed to the significant memory reduction in the universal list model compared to the classical model and the need for only local knowledge for nodes, the universal list broadcast time of all instances considered in this study is very close to that of classical; mostly between 1 to 3 time units difference.

For most instances reported in Table 8.9, HUB-GA finds a sufficiently close broadcast time under all three models ($M = \{fa, a, na\}$) compared to the best upper bound reported in the literature. This gap is even smaller for Harary networks $H_{k,n}$ and Shuffle Exchange graphs SE_d, where the

Table 8.9: Results of Experiment 4 for interconnection networks

Instance	$\|V\|$	$\|E\|$	Density	*LB* on $B_{cl}(v,G)$		*UB* on $B_{cl}(v,G)$						HUB-GA		
				TLB	LBB	TreeBlock	NTBA	NEWH	ILP	ACS	BRKGA	$B^{\sigma}_{fa}(G)$	$B^{\sigma}_{a}(G)$	$B^{\sigma}_{na}(G)$
H_2	4	4	0.6667	2	-	-	-	-	-	-	-	2	2	2
H_3	8	12	0.4285	3	-	-	-	-	-	-	-	3	4	4
H_4	16	32	0.2667	4	-	-	-	-	4	-	-	4	5	6
H_5	32	80	0.1613	5	5	5	5	5	5	5	5	5	7	9
H_6	64	192	0.0952	6	6	6	7	6	6	6	6	7	9	11
H_7	128	448	0.0551	7	7	7	9	7	7	7	7	8	11	14
H_8	256	1024	0.0314	8	8	8	11	8	8	8	9	9	13	16
H_9	512	2304	0.0176	9	9	9	14	9	9	9	10	10	15	18
CCC_2	8	12	0.4285	3	-	-	-	-	-	-	-	4	4	5
CCC_3	24	36	0.1304	5	6	-	6	7	6	6	6	8	8	10
CCC_4	64	94	0.0476	6	8	-	7	9	9	9	9	11	11	15
CCC_5	160	240	0.0189	8	10	-	11	12	11	12	11	14	15	19
CCC_6	384	576	0.0078	9	13	-	14	14	13	14	13	17	18	24
DB_3	8	16	0.5714	3	-	-	4	4	-	-	-	4	4	5
DB_4	16	32	0.2583	4	4	4	5	5	-	5	5	6	6	7
DB_5	32	64	0.1270	5	5	7	7	7	-	6	6	7	8	10
DB_6	64	128	0.0630	6	6	8	8	8	-	8	8	9	10	13
DB_7	128	256	0.0314	7	7	12	10	10	-	10	9	11	12	16
DB_8	256	512	0.0157	8	8	12	12	12	-	12	11	13	14	19
DB_9	512	1024	0.0078	9	9	14	13	13	-	14	13	15	17	22
SE_2	4	5	0.8334	2	-	-	-	-	-	-	-	3	3	4
SE_3	8	12	0.4285	3	-	-	5	5	-	-	-	5	5	6
SE_4	16	21	0.1750	4	7	-	7	7	7	7	7	7	7	9
SE_5	32	46	0.0927	5	9	-	9	9	9	9	9	9	10	13
SE_6	64	93	0.0461	6	11	-	11	11	11	11	11	12	12	16
SE_7	128	190	0.0234	7	13	-	13	13	13	13	13	15	15	20
SE_8	256	381	0.0117	8	15	-	15	15	15	15	15	17	17	25
SE_9	512	766	0.0059	9	17	-	18	18	17	17	18	20	21	28
$H_{10,30}$	30	150	0.3448	5	3	6	-	-	5	5	5	7	9	9
$H_{11,50}$	50	275	0.2245	6	3	7	-	-	6	6	6	8	10	11
$H_{20,50}$	50	500	0.4082	6	3	8	-	-	6	6	6	8	10	11
$H_{21,50}$	50	525	0.4286	6	2	7	-	-	6	6	6	7	10	10
$H_{2,100}$	100	100	0.0202	7	50	50	-	-	50	50	50	50	50	67
$H_{2,17}$	17	17	0.1250	4	8	9	-	-	9	9	9	9	9	11
$H_{2,30}$	30	30	0.0690	5	15	15	-	-	15	15	15	15	15	20
$H_{2,50}$	50	50	0.0408	6	25	25	-	-	25	25	25	25	25	29
$H_{3,17}$	17	26	0.1912	4	4	5	-	-	5	5	5	6	6	8
$H_{3,30}$	30	45	0.1034	5	8	9	-	-	9	9	9	9	9	12
$H_{3,50}$	50	75	0.0612	6	13	14	-	-	14	14	14	14	15	17
$H_{5,17}$	17	43	0.3162	4	3	5	-	-	5	5	5	5	6	7
$H_{6,17}$	17	51	0.3750	4	3	5	-	-	5	5	5	6	6	7
$H_{7,17}$	17	60	0.4412	4	2	5	-	-	5	5	5	5	6	6
$H_{8,30}$	30	120	0.2759	5	4	6	-	-	5	6	5	8	9	10
$H_{9,30}$	30	135	0.3103	5	3	6	-	-	5	5	5	7	8	9

Table 8.10: Results of Experiment 4 for connected complex networks

Instance	$\|V\|$	$\|E\|$	Density	*LB* on $B_{cl}(v,G)$		*UB* on $B_{cl}(v,G)$						HUB-GA		
				TLB	LBB	TreeBlock	NTBA	NEWH	ILP	ACS	BRKGA	$B^{\sigma}_{fa}(G)$	$B^{\sigma}_{a}(G)$	$B^{\sigma}_{na}(G)$
SW-100-3-0d1-trial1	100	100	0.0202	7	61	-	-	-	61	61	61	68	68	104
SW-100-3-0d2-trial1	100	100	0.0202	7	31	-	-	-	31	31	31	40	40	60
SW-100-3-0d2-trial3	100	100	0.0202	7	31	-	-	-	31	31	31	49	49	74
SW-100-4-0d1-trial1	100	200	0.0404	7	7	-	-	-	9	10	9	14	14	19
SW-100-4-0d1-trial2	100	200	0.0404	7	7	-	-	-	8	9	8	13	14	18
SW-100-4-0d1-trial3	100	200	0.0404	7	9	-	-	-	10	11	10	15	16	20
SW-100-4-0d2-trial1	100	200	0.0404	7	7	-	-	-	8	9	8	12	13	17
SW-100-4-0d2-trial2	100	200	0.0404	7	7	-	-	-	8	9	9	12	13	16
SW-100-4-0d2-trial3	100	200	0.0404	7	7	-	-	-	9	9	9	12	13	17
SW-100-4-0d3-trial1	100	200	0.0404	7	6	-	-	-	8	9	8	12	13	16
SW-100-4-0d3-trial2	100	200	0.0404	7	6	-	-	-	8	8	8	11	12	15
SW-100-4-0d3-trial3	100	200	0.0404	7	7	-	-	-	8	9	8	11	12	15
SW-100-5-0d1-trial1	100	200	0.0404	7	8	-	-	-	9	10	9	14	15	19
SW-100-5-0d1-trial2	100	200	0.0404	7	9	-	-	-	10	11	10	15	15	22
SW-100-5-0d1-trial3	100	200	0.0404	7	11	-	-	-	12	13	12	15	16	21
SW-100-5-0d2-trial1	100	200	0.0404	7	8	-	-	-	9	10	10	13	14	17
SW-100-5-0d2-trial2	100	200	0.0404	7	9	-	-	-	9	10	10	12	13	17
SW-100-5-0d2-trial3	100	200	0.0404	7	7	-	-	-	8	9	9	12	13	18
SW-100-5-0d3-trial1	100	200	0.0404	7	6	-	-	-	8	8	8	11	12	15
SW-100-5-0d3-trial2	100	200	0.0404	7	6	-	-	-	8	8	8	11	12	16
SW-100-5-0d3-trial3	100	200	0.0404	7	6	-	-	-	8	8	8	11	12	15
SW-100-6-0d1-trial1	100	300	0.0606	7	5	-	-	-	7	8	8	12	13	16
SW-100-6-0d1-trial2	100	300	0.0606	7	6	-	-	-	8	9	8	12	13	16
SW-100-6-0d1-trial3	100	300	0.0606	7	6	-	-	-	7	8	8	12	14	17
SW-100-6-0d2-trial1	100	300	0.0606	7	6	-	-	-	7	8	7	11	13	15
SW-100-6-0d2-trial2	100	300	0.0606	7	4	-	-	-	7	8	7	10	12	14
SW-100-6-0d2-trial3	100	300	0.0606	7	4	-	-	-	7	8	7	10	12	15
SW-100-6-0d3-trial1	100	300	0.0606	7	4	-	-	-	7	8	7	10	11	14
SW-100-6-0d3-trial2	100	300	0.0606	7	5	-	-	-	7	8	7	10	11	13
SW-100-6-0d3-trial3	100	300	0.0606	7	5	-	-	-	7	8	7	10	11	14

fully-adaptive and adaptive broadcast time is even equal to the classical upper bounds in several instances. For complex networks in Table 8.10, `HUB-GA` is able to decrease its objective function to 10, 11, and 14 for various instances under fully-adaptive, adaptive, and non-adaptive models, respectively. The value of classical broadcasting is mostly close to 8 in those instances. This result is promising because, with a careful design of the broadcast scheme under the universal list model, the best-known upper bounds reported in the literature for the classical model are achievable.

Chapter 9

Conclusion and Future work

9.1 Conclusion

A fundamental problem in the information dissemination area is broadcasting, in which a sender, usually called the $originator$, wishes to transmit a message to all network members using the communication links. At any given time, any informed network member can pass the message to one of its uninformed neighbors with a $call$. A member may receive the message from multiple senders at the same time. Finding the minimum broadcast time of a network has been proved to be NP-Hard; therefore, various broadcasting models are defined in the literature that simulates real-world situations. Broadcasting with Universal lists is one in which each vertex is equipped with a list that it must follow regardless of the originator. In this study, we focused on both classical and universal lists broadcasting.

In Chapter 3, we defined a graph structure: Fully Connected Trees FCT_n. An FCT_n is made up of trees such that the roots of the trees form a complete graph of size n. We presented a $O(|V| \log \log n)$ algorithm to calculate the exact broadcast time of any vertex in an arbitrary FCT_n. We also presented the details of the proof of correctness for our algorithm. Then in Chapter 4, we replaced the complete graph with a hypercube and proposed a heuristic for broadcasting in a hypercube of trees. The heuristic benefits from a 2-approximation ratio, which makes it theoretically significant. Besides, our numerical results indicate that the heuristic is able to outperform the best-known algorithm for the same problem in up to 90% of the experiments while speeding up the process up to 30%. Interestingly, the performance of the heuristic could be improved even if the size of the hypercube surges.

In Chapter 5, we proposed a new model, namely fully adaptive, for the problem of broadcasting with universal lists and studied its various applications in real-world networks. This model benefits from a more efficient memory complexity than the classical model. Moreover, the fully-adaptive model accelerates the broadcast process compared to adaptive and non-adaptive models. We designed optimal broadcast schemes for Grids and Cube Connected Cycles under the Fully-adaptive model. We also proved that the broadcast time of any tree under the fully-adaptive model is equal to that of the classical model. We provided an additive approximation algorithm for Tori under this newly introduced model. Finally, we introduced an infinitely large graph for which the classical broadcast time is strictly less than that of fully adaptive. In Chapter 6, we focused on the non-adaptive model of broadcasting with universal lists and proved the optimal broadcast time of complete $k-$ary trees and binomial trees under this model. Then we presented an upper bound for a complete bipartite graph and improved the general upper bound on the broadcast time of arbitrary trees. By comparing universal lists and messy broadcasting models, we proposed general upper bounds on the broadcast time of a graph G under the universal lists model. We showed that the bounds could not be improved in general. In Chapter 7, we studied *broadcast graphs (bg)'s* and *minimum broadcast graphs (mbg)'s* under the fully-adaptive model. In particular, we presented mbg's on n vertices for $n \leq 10$ and sparse bg's for $11 \leq n \leq 14$. We also suggested four infinite families of broadcast graphs under the fully-adaptive model using a general construction. Lastly, we proved that Hypercube H_d is an mbg for any value of $n = 2^d$ under the fully-adaptive model.

Finally, in Chapter 8, we proposed `HUB-GA`: a Heuristic for the problem of Universal lists Broadcasting that uses Genetic Algorithm. Our numerical results have demonstrated that our algorithm is able to find optimal or near-optimal broadcast time for several well-known interconnection networks. It also outperforms degree-based heuristics for various networks with clique-like subgraphs, in which the problem's search space is almost maximized. As opposed to the significant memory reduction in the universal list model compared to the classical model, our result is very close to the state-of-the-art heuristics for the classical broadcast problem for interconnection networks and several instances of synthetic networks.

9.2 Future Work

In what follows, we shed some light on possible future works for each Chapter of this study:

- For Chapter 3, studying FCT_n is a step toward studying graphs containing subgraphs with well-known graph topologies. In an FCT_n, the subgraphs were trees and one complete graph. More complicated graph structures can be obtained by having different subgraphs. We believe these results can be helpful when studying the broadcast problem in more complex graph structures and can also be useful in designing approximation algorithms if an exact algorithm cannot be obtained. Another future work will be to close the gap between the obvious lower bound $\Omega(|V|)$ and the current algorithm $O(|V| \log \log n)$.

- Considering Chapter 4, studying the performance of the heuristic in real-world data sets would be interesting. Besides, the approximability of this problem could be analyzed by trying to either come up with an additive approximation or proving the NP-Hardness of this problem. Additionally, one may replace the hypercube with any other class of graphs in which the broadcast scheme and broadcast time are known. Some examples include Grids, Tori, Cube-Connected Cycles, and Shuffle Exchange graphs.

- For Chapter 5, since we introduced a new model, we leave open several exciting research questions. For instance, the broadcast time of various fundamental families of networks, such as Shuffle Exchange, DeBruijn, and Harary graphs, is still unknown under the fully-adaptive model. We believe the current upper bound is not tight for complete graphs and can be improved. Also, studying the widest margin between a graph's classical and fully-adaptive broadcast time on n vertices could be an interesting theoretical question. Finally, a significant breakthrough in this area is to design an efficient approximation/heuristic algorithm to find $B_{fa}(G)$ for general graphs or prove that it is NP-Hard.

- For Chapter 6, the non-adaptive broadcast time of several interconnection networks is still unknown, or the current upper bound is not tight. Examples include Hypercubes, Cube Connected Cycles, DeBruijn, Shuffle Exchange, and Harary graphs. We think the general lower bounds on trees could be improved by carefully considering the degrees of diametral vertices.

- For Chapter 7, the problem of finding mbg's and bg's for greater values of n under the fully-adaptive model is still widely open. The answer to the following question is also interesting: "is there any value of n, for which $B^{(cl)}(n) < B^{(fa)}(n)$?" We conjecture that Knödel graph KG_{2^k-2} might be an mbg under the fully-adaptive model for $n = 2^k - 2$. Besides, bg's are not even defined for the adaptive and non-adaptive models where the achievability of the obvious lower bound of $\lceil \log n \rceil$ is questionable. Thus, by formalizing this problem for these setups, the problem of finding bg's and mbg's for these two models could gain attention in the future.

- For Chapter 8, one immediate direction is the improvement of `HUB-GA` using various techniques of genetic algorithms. Also, the proposed algorithm still needs to be examined for diverse real-world networks. Moreover, other evolutionary algorithms, such as ant colony and particle swarm optimization, could be utilized instead of the Genetic Algorithm. Finally, the idea used in our approach could be extended to classical broadcasting to minimize the broadcast time of the graph, not a particular vertex. In that case, it could be the first heuristic that minimizes the broadcast time of the graph while giving the actual broadcast scheme under the classical model.

9.3 Publications

This Ph.D. degree resulted in the following scientific productions:

(1) Chapter 3: (Gholami, Harutyunyan, & Maraachlian, 2022).

(2) Chapter 4: (Gholami & Harutyunyan, 2021).

(3) Chapter 5: (Gholami & Harutyunyan, 2022b).

(4) Chapter 6: (Gholami & Harutyunyan, 2022d).

(5) Chapter 7: (Gholami & Harutyunyan, 2022a).

(6) Chapter 8: (Gholami & Harutyunyan, 2022c).

(7) In collaboration with other researchers: (Bakhtar, Gholami, & Harutyunyan, 2020).

(8) In collaboration with other researchers: (Gholami, Saghiri, Vahidipour, & Meybodi, 2021).

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.

Ahn, C. W., & Ramakrishna, R. S. (2003). Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation*, *7*(4), 367–385.

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

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.

Al-Jawad, A., Comşa, I.-S., Shah, P., Gemikonakli, O., & Trestian, R. (2021). An innovative reinforcement learning-based framework for quality of service provisioning over multimedia-based SDN environments. *IEEE Transactions on Broadcasting*, *67*(4), 851–867.

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.

Assi, M., Halawi, B., & Haraty, R. A. (2018). Genetic algorithm analysis using the graph coloring method for solving the university timetable problem. *Procedia Computer Science*, *126*, 899–906.

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., 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.

Baroudi, U., Bin-Yahya, M., Alshammari, M., & Yaqoub, U. (2019). Ticket-based QoS routing optimization using genetic algorithm for WSN applications in smart grid. *Journal of Ambient Intelligence and Humanized Computing*, *10*(4), 1325–1338.

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.

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).

Bermond, J.-C. (1979). Graceful graphs, radio antennae and french windmills. In *Proceedings One day Combinatorics Conference, Research Notes in Mathematics* (pp. 18–37).

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

Bermond, J.-C., Hell, P., Liestman, A. L., & Peters, J. G. (1992). 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.

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., Harutyunyan, H., & 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. (2014). Broadcast problem in hypercube of trees. In *International Workshop on Frontiers in Algorithmics* (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., & Tanna, S. (2014). Broadcasting in Harary-Like Graphs. In *2014 IEEE 17th International Conference on Computational Science and Engineering* (p. 1269-1276).

Bhardwaj, A., & El-Ocla, H. (2020). Multipath routing protocol using genetic algorithm in mobile ad hoc networks. *IEEE Access*, *8*, 177534–177548.

Bhola, J., Soni, S., & Cheema, G. K. (2020). Genetic algorithm based optimized LEACH protocol for energy efficient wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing*, *11*(3), 1281–1288.

Binh, L. H., & Duong, T. V. T. (2021). Load balancing routing under constraints of quality of transmission in mesh wireless network based on software defined networking. *Journal of Communications and Networks*, *23*(1), 12-22.

Bucantanschi, D., Hoffmann, B., Hutson, K. R., & Kretchmar, R. M. (2007). A neighborhood search technique for the freeze tag problem. In *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies* (pp. 97–113). Springer.

Cabral, R. S., Aquino, A., Frery, A., Rosso, O., & Ramírez, J. (2013). Structural changes in data communication in wireless sensor networks. *Open Physics*, *11*(12), 1645–1652.

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.

Chu, C.-H., Premkumar, G., & Chou, H. (2000). Digital data networks design using genetic algorithms. *European Journal of Operational Research*, *127*(1), 140–158.

Chu, X., & Chen, Y. (2018). Time division inter-satellite link topology generation problem: Modeling and solution. *International Journal of Satellite Communications and Networking*, *36*(2), 194–206.

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.

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., Rodrıguez-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.

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* (pp. 152–158).

Elkin, M., & Kortsarz, G. (2005). A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem. *SIAM Journal on Computing*, *35*(3), 672–689.

Elkin, M., & Kortsarz, G. (2006). Sublogarithmic approximation for telephone multicast. *Journal of Computer and System Sciences*, *72*(4), 648–659.

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).

Fang, Y., & Li, J. (2010). A review of tournament selection in genetic programming. In *International Symposium on Intelligence Computation and Applications* (pp. 181–192).

Farley, A. M. (1979). Minimal broadcast networks. *Networks*, *9*(4), 313–332.

Farley, A. M. (2004). Minimal path broadcast networks. *Networks*, *43*(2), 61–70.

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.

Fortez, G., Robledo, F., Romero, P., & Viera, O. (2020). A fast genetic algorithm for the max cut-clique problem. In *International Conference on Machine Learning, Optimization, and Data Science* (pp. 528–539).

Fraigniaud, P., & Lazard, E. (1994). Methods and problems of communication in usual networks. *Discrete Applied Mathematics*, *53*(1), 79-133.

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.

Freitas, C. G., Aquino, A. L., Ramos, H. S., Frery, A. C., & Rosso, O. A. (2019). A detailed characterization of complex networks using Information Theory. *Scientific Reports*, *9*(1), 1–12.

Garey, M. R., & Johnson, D. S. (1983). *Computers and intractability. A guide to the theory of NP-Completeness.*

Gargano, L., & Vaccaro, U. (1989). On the construction of minimal broadcast networks. *Networks*, *19*(6), 673–689.

Gholami, S., & Harutyunyan, H. A. (2021). A Broadcasting Heuristic for Hypercube of Trees. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0355–0361).

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).

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. (2022d). A Note on Non-adaptive Broadcasting with Universal Lists. *Special issue on Graph and Combinatorial Optimization for Big Data Intelligence with Parallel Processing, Parallel Processing Letters (Under Review).*

Gholami, S., Harutyunyan, H. A., & Maraachlian, E. (2022). Optimal Broadcasting in Fully Connected Trees. *Journal of Interconnection Networks*, 2150037.

Gholami, S., Saghiri, A. M., Vahidipour, S., & Meybodi, M. (2021). HLA: a novel hybrid model based on fixed structure and variable structure learning automata. *Journal of Experimental & Theoretical Artificial Intelligence*, 1–26.

Goldberg, D. E. (1990). A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, *4*, 445–460.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms* (Vol. 1, pp. 69–93). Elsevier.

Grefenstette, J. J. (1993). Genetic algorithms and machine learning. In *Proceedings of the sixth Annual Conference on Computational learning theory* (pp. 3–4).

Grigni, M., & Peleg, D. (1991). Tight bounds on mimimum broadcast networks. *SIAM Journal on Discrete Mathematics*, *4*(2), 207–222.

Guidoni, D. L., Mini, R. A., & Loureiro, A. A. (2010). On the design of resilient heterogeneous wireless sensor networks based on small world concepts. *Computer Networks*, *54*(8), 1266–1281.

Harary, F. (1962). The maximum connectivity of a graph. *Proceedings of the National Academy of Sciences*, *48*(7), 1142–1146.

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. (2006). 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., & 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., & 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., & Liestman, A. L. (1998). Messy broadcasting. *Parallel Processing Letters*, *8*(02), 149–159.

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., Makino, K., & Shermer, T. C. (2011). Nonadaptive broadcasting in trees. *Networks*, *57*(2), 157–168.

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.

Harutyunyan, H. A., & Taslakian, P. (2004). Orderly broadcasting in a 2d torus. In *Proceedings Eighth International Conference on Information Visualisation* (pp. 370–375).

Harutyunyan, H. A., & Wang, W. (2010). Broadcasting algorithm via shortest paths. In *2010 IEEE 16th International Conference on Parallel and Distributed Systems* (pp. 299–305).

Hasson, Y., & Sipper, M. (2004). A novel ant algorithm for solving the minimum broadcast time problem. In *International Conference on Parallel Problem Solving from Nature* (pp. 501–510).

Hedetniemi, S. M., Hedetniemi, S. T., & Liestman, A. L. (1988). A survey of gossiping and broadcasting in communication networks. *Networks*, *18*(4), 319–349.

Ho, C. T., & Johnsson, S. L. (1986). Distributed routing algorithms for broadcasting and personalized communication in hypercubes. In *ICPP* (pp. 640–648).

Hocaoğlu, M. F., & Genç, I. (2019). Smart combat simulations in terms of industry 4.0. In *Simulation for Industry 4.0* (pp. 247–273). Springer.

Hoelting, C. J., Schoenefeld, D. A., & Wainwright, R. L. (1996). A genetic algorithm for the minimum broadcast time problem using a global precedence vector. In *Proceedings of the 1996 ACM symposium on Applied Computing* (pp. 258–262).

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

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.

Hutter, M. (2002). Fitness uniform selection to preserve genetic diversity. In *Proceedings of the 2002 Congress on Evolutionary Computation* (Vol. 1, pp. 783–788).

Ishibuchi, H., & Yamamoto, T. (2004). Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy sets and Systems*, *141*(1), 59–88.

Ivanova, M. (2019). Optimization problems in communication networks and multi-agent path finding. *Ph.D. Thesis, The University of Bergen*.

Jakoby, A., Reischuk, R., & Schindelhauer, C. (1998). The complexity of broadcasting in planar and decomposable graphs. *Discrete Applied Mathematics*, *83*(1-3), 179–206.

Jiao, Y., & Joe, I. (2016). Energy-efficient resource allocation for heterogeneous cognitive radio network based on two-tier crossover genetic algorithm. *Journal of Communications and Networks*, *18*(1), 112-122.

Kamiński, B., Prałat, P., & Théberge, F. (2021). *Mining complex networks*. Chapman and Hall/CRC.

Karyotis, V., & Khouzani, M. (2016). *Malware diffusion models for modern complex networks: theory and applications*. Morgan Kaufmann.

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, *80*(5), 8091–8126.

Khachatrian, L., & Harutounian, O. (1990). Construction of new classes of minimal broadcast networks. In *Conference on Coding Theory, Armenia* (pp. 69–77).

Kim, J.-H., & Chwa, K.-Y. (2005). Optimal broadcasting with universal lists based on competitive analysis. *Networks*, *45*(4), 224–231.

Kim, Y.-H., Yoon, Y., & Geem, Z. W. (2019). A comparison study of harmony search and genetic algorithm for the max-cut problem. *Swarm and Evolutionary Computation*, *44*, 130–135.

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.

König, J.-C., & Lazard, E. (1994). Minimum k-broadcast graphs. *Discrete Applied Mathematics*, *53*(1-3), 199–209.

Kortsarz, G., & Peleg, D. (1995). Approximation algorithms for minimum-time broadcast. *SIAM Journal on Discrete Mathematics*, *8*(3), 401–427.

Kumar, A., Dadheech, P., Kumari, R., & Singh, V. (2019). An enhanced energy efficient routing protocol for VANET using special cross over in genetic algorithm. *Journal of Statistics and Management Systems*, *22*(7), 1349–1364.

Kumar, R. (2012). Blending roulette wheel selection & rank selection in genetic algorithms. *International Journal of Machine Learning and Computing*, *2*(4), 365–370.

Labahn, R. (1994). A minimum broadcast graph on 63 vertices. *Discrete Applied Mathematics*, *53*(1-3), 247–250.

Lambora, A., Gupta, K., & Chopra, K. (2019). Genetic algorithm a literature review. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)* (pp. 380–384).

Lee, J.-K., Hong, T., & Li, G. (2021). Traffic and overhead analysis of applied pre-filtering ACL firewall on HPC service network. *Journal of Communications and Networks*, *23*(3), 192-200.

Leighton, F. T. (2014). *Introduction to parallel algorithms and architectures: Arrays· trees· hypercubes*. Elsevier.

Li, C., Hart, T. E., Henry, K. J., & Neufeld, I. A. (2008). Average-case “messy” broadcasting. *Journal of Interconnection Networks*, *9*(04), 487–505.

Li, D., Wang, S., Zhu, K., & Xia, S. (2017). A survey of network update in SDN. *Frontiers of computer science*, *11*(1), 4–12.

Li, M.-s., Xue, J.-m., & Liu, Y. (2019). Research and application of bipartite graph optimal-matching algorithm on task-driven teaching method. In *The International Conference on Cyber Security Intelligence and Analytics* (pp. 885–892).

Liestman, A. L., & Peters, J. G. (1988). Broadcast networks of bounded degree. *SIAM journal on Discrete Mathematics*, *1*(4), 531–540.

Lima, A., Aquino, A. L., Nogueira, B., & Pinheiro, R. G. (2022). A matheuristic approach for the minimum broadcast time problem using a biased random-key genetic algorithm. *International Transactions in Operational Research*.

Liu, Z., Huang, L., Li, B., & Ji, B. (2021). Anti-aging scheduling in single-server queues: A systematic and comparative study. *Journal of Communications and Networks*, *23*(2), 91-105.

Louis, S. J., & Rawlins, G. J. (1991). Designer Genetic Algorithms: Genetic Algorithms in Structure Design. In *ICGA* (pp. 53–60).

Maheo, M., & Saclé, J.-F. (1994). Some minimum broadcast graphs. *Discrete Applied Mathematics*, *53*(1-3), 275–285.

Maini, H., Mehrotra, K., Mohan, C., & Ranka, S. (1994). Genetic algorithms for graph partitioning and incremental graph partitioning. In *Proceedings of the 1994 ACM/IEEE conference on Supercomputing* (pp. 449–457).

Marappan, R., & Sethumadhavan, G. (2018). Solution to graph coloring using genetic and tabu search procedures. *Arabian Journal for Science and Engineering*, *43*(2), 525–542.

Marchiori, E. (1998). A simple heuristic based genetic algorithm for the maximum clique problem. In *Symposium on Applied Computing: Proceedings of the 1998 ACM symposium on Applied Computing* (Vol. 27, pp. 366–373).

Mazaideh, M. A., & Levendovszky, J. (2021). A multi-hop routing algorithm for WSNs based on compressive sensing and multiple objective genetic algorithm. *Journal of Communications and Networks*, *23*(2), 138-147.

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.

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, *9*(3), 193–212.

Mirjalili, S. (2019). Genetic algorithm. In *Evolutionary algorithms and neural networks* (pp. 43–55). Springer.

Mitchell, S., & Hedetniemi, S. (1980). A census of minimum broadcast graphs. *Journal of Combinatorics, Information, and System Sciences*, *5*, 141–151.

Morosan, C. D. (2006). On the number of broadcast schemes in networks. *Information Processing Letters*, *100*(5), 188–193.

Muruganantham, N., & El-Ocla, H. (2020). Routing using genetic algorithm in a wireless sensor network. *Wireless Personal Communications*, *111*(4), 2703–2732.

Ohsita, Y., Ata, S., & Murata, M. (2004). Detecting distributed Denial-of-Service attacks by analyzing TCP SYN packets statistically. In *IEEE Global Telecommunications Conference (GLOBECOM)* (Vol. 4, pp. 2043–2049).

Ostrouchov, G. (1987). Parallel computing on a hypercube: an overview of the architecture and some applications. In *Computer Science and Statistics, Proceedings of the 19th Symposium on the Interface* (pp. 27–32).

Palmer, C. C., & Kershenbaum, A. (1994). Representing trees in genetic algorithms. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence* (pp. 379–384).

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).

Patel, J. K., & Read, C. B. (1996). *Handbook of the normal distribution* (Vol. 150). CRC Press.

Pinto, B. Q., Ribeiro, C. C., Rosseti, I., & Plastino, A. (2018). A biased random-key genetic algorithm for the maximum quasi-clique problem. *European Journal of Operational Research*, *271*(3), 849–865.

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).

Rezaeipanah, A., Nazari, H., & Ahmadi, G. (2019). A hybrid approach for prolonging lifetime of wireless sensor networks using genetic algorithm and online clustering. *J. Comput. Sci. Eng.*, *13*(4), 163–174.

Rhodes, M. L., & Wong, T. G. (2019). Quantum walk search on the complete bipartite graph. *Physical Review A*, *99*(3), 032301.

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).

Rocher-Gonzalez, J., Escudero-Sahuquillo, J., García, P. J., & Quiles, F. J. (2017). On the impact of routing algorithms in the effectiveness of queuing schemes in high-performance interconnection networks. In *IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI)* (pp. 65–72).

Rosenthal, A., & Scheuermann, P. (1987). Universal rankings for broadcasting in tree networks. In *Proceedings of the 25th Allerton Conference on Communication, Control and Computing* (pp. 641–649).

Rossi, R., & Ahmed, N. (2015). The network data repository with interactive graph analytics and visualization. In *Twenty-ninth AAAI Conference on Artificial Intelligence.*

Saclé, J.-F. (1996). Lower bounds for the size in four families of minimum broadcast graphs. *Discrete Mathematics*, *150*(1-3), 359–369.

Safe, M., Carballido, J., Ponzoni, I., & Brignole, N. (2004). On stopping criteria for genetic algorithms. In *Brazilian Symposium on Artificial Intelligence* (pp. 405–413).

Salkuyeh, M. A., & Abolhassani, B. (2018). Optimal video packet distribution in multipath routing for urban VANETs. *Journal of Communications and Networks*, *20*(2), 198-206.

Scheuermann, P., & Wu, G. (1984). Heuristic algorithms for broadcasting in point-to-point computer networks. *IEEE Computer Architecture Letters*, *33*(09), 804–811.

Scott-Hayward, S., O'Callaghan, G., & Sezer, S. (2013). SDN security: A survey. In *2013 IEEE SDN For Future Networks and Services (SDN4FNS)* (pp. 1–7).

Selvanathan, N., & Tee, W. J. (2003). A genetic algorithm solution to solve the shortest path problem in OSPF and MPLS. *Malaysian Journal of Computer Science*, *16*(1), 58–67.

Semenkin, E., & Semenkina, M. (2012). Self-configuring genetic algorithm with modified uniform crossover operator. In *International Conference in Swarm Intelligence* (pp. 414–421).

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., … Rao, N. (2013). Are we ready for SDN? implementation challenges for software-defined networks. *IEEE Communications magazine*, *51*(7), 36–43.

Shang, W., Wan, P., & Hu, X. (2010). Approximation algorithms for minimum broadcast schedule problem in wireless sensor networks. *Frontiers of Mathematics in China*, *5*(1), 75–87.

Shanmugasundaram, N., Sushita, K., Kumar, S. P., & Ganesh, E. (2019). Genetic algorithm-based road network design for optimising the vehicle travel distance. *International Journal of Vehicle Information and Communication Systems*, *4*(4), 344–354.

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.

Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. *Computer*, *27*(6), 17–26.

Sun, B., & Li, L. (2006). A QoS multicast routing optimization algorithm based on genetic algorithm. *Journal of Communications and Networks*, *8*(1), 116-122.

Talbi, E.-G., & Bessiere, P. (1991). A parallel genetic algorithm for the graph partitioning problem. In *Proceedings of the 5th International Conference on Supercomputing* (pp. 312–320).

Tsutsui, S., Yamamura, M., & Higuchi, T. (1999). Multi-parent recombination with simplex crossover in real coded genetic algorithms. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation* (pp. 657–664).

Varvarigos, E. A., & Bertsekas, D. P. (1995). Dynamic broadcasting in parallel computing. *IEEE Transactions on Parallel and Distributed Systems*, *6*(2), 120–131.

Vissicchio, S., Vanbever, L., Cittadini, L., Xie, G. G., & Bonaventure, O. (2017). Safe update of hybrid SDN networks. *IEEE/ACM Transactions on Networking*, *25*(3), 1649–1662.

Wang, T., Zhang, G., Yang, X., & Vajdi, A. (2018). Genetic algorithm for energy-efficient clustering and routing in wireless sensor networks. *Journal of Systems and Software*, *146*, 196–214.

Wen, X., Ruan, Y., Li, Y., Xia, H., Zhang, R., Wang, C., … Jiang, X. (2022). Improved genetic algorithm based 3-D deployment of UAVs. *Journal of Communications and Networks*, *24*(2), 223-231.

Wu, Y., & Liu, W. (2013). Routing protocol based on genetic algorithm for energy harvesting-wireless sensor networks. *IET Wireless Sensor Systems*, *3*(2), 112–118.

Xiao, J., & Wang, X. (1988). A research on minimum broadcast graphs. *Chinese Journal of Computers*, *11*, 99–105.

Yang, J.-S., Chan, H.-C., & Chang, J.-M. (2011). Broadcasting secure messages via optimal independent spanning trees in folded hypercubes. *Discrete Applied Mathematics*, *159*(12), 1254–1263.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, *33*(4), 452–473.

Zbigniew, M. (1996). Genetic algorithms+ data structures= evolution programs. In *Computational Statistics* (pp. 372–373). Springer-Verlag.

Zhang, D.-g., Liu, S., Liu, X.-h., Zhang, T., & Cui, Y.-y. (2018). Novel dynamic source routing protocol (DSR) based on genetic algorithm-bacterial foraging optimization (GA-BFO). *International Journal of Communication Systems*, *31*(18), e3824.

Zhang, G., Wu, M., Duan, W., & Huang, X. (2018). Genetic algorithm based QoS perception routing protocol for VANETs. *Wireless Communications and Mobile Computing*, *2018*.

Zhou, J.-g., & Zhang, K.-m. (2001). A minimum broadcast graph on 26 vertices. *Applied Mathematics Letters*, *14*(8), 1023–1026.