OPTIMAL BROADCASTING IN TREELIKE GRAPHS

Edward Maraachlian

A thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

May 2010

Your file Votre référence
ISBN: 978-0-494-71135-4
Our file Notre référence
ISBN: 978-0-494-71135-4

Abstract

Optimal Broadcasting in Treelike Graphs

Edward Maraachlian, Ph.D.

Concordia University, 2010

Broadcasting is an information dissemination problem in a connected network, in which one node, called the *originator*, disseminates a message to all other nodes by placing a series of calls along the communication lines of the network. Once informed, the nodes aid the originator in distributing the message. Finding the broadcast time of a vertex in an arbitrary graph is NP-complete. The problem is solved polynomially only for a few classes of graphs. In this thesis we study the broadcast problem in different classes of graphs which have various similarities to trees. The unicyclic graph is the simplest graph family after trees, it is a connected graph with only one cycle in it. We provide a linear time solution for the broadcast problem in unicyclic graphs. We also studied graphs with increasing number of cycles and complexity and provide again polynomial time solutions. These graph families are: tree of cycles, necklace graphs, and 2-restricted cactus graphs. We also define the fully connected tree graphs and provide a polynomial solution and use these results to obtain polynomial solution for the broadcast problem in tree of cliques and a constant approximation algorithm for the hierarchical tree cluster networks.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Hovhanness Harutyunyan who helped, guided, and motivated me since the first day that I arrived at Concordia.

Contents

List of Figures

Chapter 1

Introduction

Even though modern day CPUs are getting faster every day, they are still unable to solve a plethora of problems that scientists face. One of the shortcomings of single CPU systems is the long amount of time needed to solve the problem serially. A second shortcoming is that sometimes problems would not even fit in the memory of a single CPU system. The answer to these shortcomings is parallelism. Multiple CPUs working on the same problem often can solve the problems more quickly. Parallel computing is the term used to describe the usage of mutiple CPUs to solve a single problem. Parallel computing, once only used by scientists and engineers in expensive computer labs, is becoming the default processing environment used by common people. The current generation of processors are all multi-core having several processing units on the same board. These systems, however, share the same memory and hence might face the second problem mentioned above: inability to fit the problem in the memory.

There are different models for parallel computing. One of the most common models is the MIMD (Multiple Instruction and Multiple Data) which is sometimes referred to multicomputers or multiprocessors. The different processors, working in parallel, will most probably need to exchange data among each other. This is done either through a shared memory or an interconnection network. Shared memory multicomputers have a limitation on the number of processors that can be connected together. Hence, it is not practical if a very large number of processors is to be connected. A more realistic way of designing multicomputers is to make each processor have its own main memory. Communication between the processors will be accomplished by passing messages using an interconnection network. It turns out that the performance of these multicomputers not only depends on the processing power of the processors but also on the performance of the interconnection network in disseminating data among the processor. Research has shown that the structural properties of a network determine many of its characteristics such as the minimum communication time, ease of routing, and fault tolerance.

One of the fundamental information dissemination problems is broadcasting. Broadcasting is a process in which a single message is sent from one member of a network to all other members. Inefficient broadcasting could degrade the performance of a network seriously. Therefore, it is of a major interest to improve the performance of a network by using efficient broadcasting algorithms.

Broadcasting is an information dissemination problem in a connected network, in which one node, called the *originator*, must distribute a message to all other nodes by placing a series of calls along the communication lines of the network. Once informed, the informed nodes aid the originator in distributing the message. This is assumed to take place in discrete time units. The broadcasting is to be completed as quickly as possible, subject to the following constraints:

- Each call involves only one informed node and one of its uninformed neighbors.
- Each call requires one unit of time.
- A node can participate in only one call per unit of time.
- In one unit of time, many calls can be performed in parallel.

A *broadcast scheme* of an originator u is a set of calls that, starting at vertex u, completes the broadcasting in the network.

Formally, any network can be modelled as a connected graph $G = (V, E)$, where V is the set of vertices (or nodes) and E is the set of edges (or communication lines) between the vertices in graph G.

Given a connected graph $G = (V, E)$ and a message originator, vertex u, the *broadcast time* of vertex u, $b(u, G)$ or $b(u)$, is the minimum number of time units required to complete broadcasting from the vertex u. Note that for any vertex u in a connected graph G on n vertices, $b(u) \geq \lceil \log n \rceil$ since during each time unit the number of informed vertices can at most be doubled. On the other hand in a connected graph there should be at least one new informed vertex at every new round which implies that $b(u) \leq n - 1$. The broadcast time $b(G)$ of the graph G is defined as $max\{b(u)|u \in V\}$. The edges that get used during a broadcast process form a spanning tree of the graph. This spanning tree is called a broadcast tree. For surveys of results on broadcasting and related problems, see Hedetniemi, Hedetniemi, and Liestman [78], Fraigniaud and Lazard [43], Hromkovic, Klasing, Monien, and Peine [81], Hromkovic, Klasing, Pelc, Ruzicka, and Unger [84].

Determination of $b(u, G)$ or $b(u)$ for a vertex u in an arbitrary graph G is NP-complete [87]. The proof of NP-completeness is presented in [112]. Therefore, many papers have presented approximation or heuristic algorithms to determine the broadcast time of a vertex u in G, $b(u, G)$ (see [5, 32, 33, 38, 44, 46, 98, 110, 10, 45, 113, 75]).

Since the problem is NP-complete in general, another direction for research is to design polynomial algorithms that determine the broadcast time of any vertex for a class of graphs. The broadcast problem remains NP-complete even in some restricted classes of graphs namely, planar graphs and bounded degree graphs [85]. The same paper also states that graphs with certain decomposition properties have a polynomial solution. The first class of graphs to be studied and found to have a linear solution is the trees [112]. The authors propose an algorithm that finds the broadcast time of a given vertex in an arbitrary tree. They also find a broadcast scheme of a given tree $T = (V, E)$ in linear time $O(|V|)$. Other graph classes where the the broadcast problem was studied are the hypercube, cube connected cycles, butterfly graph, shuffle exchange and the de Bruijn networks.

The rest of this thesis is organized as follows. In the next chapter we present a literature review of some of the important results on the broadcast problem in general and the different network classes that have been considered. In the third chapter we will present our results on unicyclic graphs which are connected graphs with n edges and on n vertices. In Chapter 4 we study the broadcast problem in graphs which are subfamilies of the cactus graphs. In Chapter 5 we introduce a class of graphs which we call fully connected trees and study the broadcast problem in these graphs. In Chapter 6 we define a graph family which we call hierarchical tree cluster networks. We present an exact broadcast algorithm for the cases where the clusters are cliques and present an approximation algorithm for more general structures of the clusters. Finally, Chapter 7 is the conclusion and a short note on future work.

Chapter 2

Literature Review

In this chapter we review the major results of the broadcast problem. In this thesis we will consider the classical model of broadcasting known as the telephone model. In this model a node in the network can communicate with only one of its neighbors and both vertices can exchange all the information they have, this is sometimes called full-duplex communication mode. Sometimes broadcast models are categorized based on the number of neigbhors a processor can communicate with simoultaneously. In the 1-port communication model, a processor can communicate with only one neighbor at a time. This model is sometimes refered to as the processor bound model. The other extreme, the link bound model, is the case where a processor can send an information to all of its neighbors at the same time. The k-port broadcast [57, 58, 62] model is something in the middle of the two previous models where a processor can communicate with at most k of its neigbhours at a time.

Another differentiator between broadcast models is the assumption about the time needed to send a message between two nodes on the network. The time needed is actually the total time which includes the time to prepare a message for sending, the time needed by the message to propagate from one node to another, and the time needed by the receiver to physically receive the message. There are 2 widely different models that are actually used.

1. The constant model, where the time needed to send a message from one node to another is constant regardless of the size of the message.

2. The linear model, where the time needed to communicate a message between two neighboring nodes is a linear function of the size of the message.

Most models in the literature use the constant model however, there are some results which deal with the linear model [109, 9, 115].

In the line broadcast model a vertex can send a message to vertices that are not its immediate neighbor. In every round a set of paths will be used to inform destination vertices. Intermediate vertices falling on the path between a source and destination vertices can either learn the message or just help in the message transmission without actually reading it. Variations of this model include the vertex disjoint path mode where the set of paths in a certain round do not have a common vertex [35, 39, 42, 82], and the edge disjoint path mode where the paths do not have a common edge [35, 39, 83, 81].

In the universal list broadcast model [26, 88] every vertex has an ordered list of neighbors that it informs in the prescribed order everytime it gets informed. This can be contrasted with the classical model where every vertex chooses the ordered list of vertices that it has to forward the message to depending on the source vertex. So in the classical model, every vertex has many lists corresponding to the different possible sources. In [60] a polynomial algorithm to determine the universal list broadcast time of a tree is presented. The universal list model implies that every node needs to have a small memory and computing power because the message forwarding schedule is unique for all originators. Note that this model is sometimes refered to as orderly broadcasting as in [76].

Another variant of broadcast models is the messy broadcasting. This model was introduced in [2]. In messy broadcasting each vertex sends the message randomly to its neighbors without knowing who the originator is or the time at which the message was sent. The vertices do not know the network topology apart from their immediate neighbors. Usually in messy broadcasting the worst case performance of a broadcast protocol is considered [55, 23, 61]. A study for average case time of messy broadcasting is done in [103].

When there is the need to communicate large amounts of data, some systems break up the information into smaller pieces which are sent individually over the network. This motivates the study of multiple message broadcasting. In this model, the originator has m messages which need to be communicated with all the n vertices of the graph. There are numerous papers dedicated to studying how to efficiently complete the broadcasting of multiple messages [7, 8, 20, 21, 36, 54, 53, 105, 25, 97, 94].

There are more generalized models too which are more applicable. In [5, 6] the basic broadcast model is generalized to obtain a model in which each node has a different switching time between messages. Good approximation algorithms are presented in [92, 93, 96].

A similar model is studied in [74, 73, 72]. Another model is the fault tolerant broadcasting where it is assumed that some links in the network can be faulty. A k fault-tolerant broadcasting scheme is a broadcast protocol that assures that any node in the network will receive the message from the originator in presence of up to k edge failures [1, 48, 47, 108].

There are numerous other broadcast models different than what was mentioned above. However, for the purpose of this thesis we will consider the classical broadcast model. In this model, the problem of finding the broadcast time of an arbitrary vertex in an arbitrary graph was proved to be NP complete. The proof was done by reducing the the well known 3-dimensional matching (3DM) problem into the broadcast problem [112]. The problem stays NP-complete even in more restricted classes of graphs such as planar graphs [85, 86] and bounded degree graphs [15, 28, 107].

Since finding an optimal broadcast scheme for general graphs was proved to be computationally very expensive, research in this area focused on the following main problems:

1. Finding the broadcast time and scheme for different classes of graphs.

2. Finding good approximation algorithms and efficient broadcast heuristics for general graphs.

3. Constructing graphs whose broadcast time is $\lceil \log n \rceil$. These graphs are called broadcast graphs, bg. This is not necessarily a difficult task if there were no cost concerns. For example a complete graph has broadcast time of $\lceil \log n \rceil$ but has more edges than what is necessary. The research in this area is directed towards constructing graphs on n vertices with broadcast time of $\lceil \log n \rceil$ and minimum number of edges. These

graphs are called minimum broadcast graphs, *mbg*.

2.1 Broadcast Time of Different Topologies

In this section we will present some of the well known graph topologies and their broadcast times. We will start with the path topology, arguably the simplest graph topology, and gradually consider more and more complex structures.

2.1.1 Path P_n

A path P_n on n vertices numbered v_1 to v_n has $n-1$ edges and every vertex v_i, $2 \leq i \leq n-1$, has two incident edges connecting it to the vertices v_{i-1} and v_{i+1}. The broadcast time of P_n is $n-1$ and this is because the end vertices have the maximum broadcast time in a path which is equal to $n-1$.

2.1.2 Cycle C_n

The cycle (ring) C_n on n vertices is a path P_n where the end vertices v_1 and v_n are also connected by an edge. $b(C_n) = \lceil \frac{n}{2} \rceil$

2.1.3 Tree T

The tree T on n vertices is a connected graph with n vertices and $n-1$ edges. There is exactly one unique path between every two vertices of a tree. The broadcast problem in general trees has been solved in [112] where a linear algorithm was presented which can find the optimal broadcast scheme of any vertex in an arbitrary tree.

2.1.4 Complete Graph K_n

The complete graph (clique) K_n on n vertices is a graph where every vertex has an edge to each of the remaining $n-1$ vertices. Hence, the number of edges of a K_n is $\frac{n(n-1)}{2}$. It can be easily seen that $b(K_n) = \lceil \log n \rceil$ because at every round, except the last one, the number of informed vertices can double.

2.1.5 Hypercube H_n

The n-dimensional hypercube, H_n, is defined to be a graph on 2^n vertices. Each vertex corresponds to an n-bit binary string, and two vertices are linked with an edge if and only if their binary strings differ in precisely one bit. For example the vertices v_2 and v_6 in H_3 are neighbours because the binary representations 010 and 110 , of 2 and 6 respectively, differ only in the third position. The hypercube is one of the few infinite family of graphs where the broadcast time is equal to $\log n$, i.e. $b(H_n) = n$.

2.1.6 Cube-Connected Cycles CCC_m

The m-dimensional cube-connected cycles, CCC_m, is a graph $G = (V_m, E)$ where $V_m = \{0, 1, \cdots, m-1\} \times \{0,1\}^m$ and $\{0,1\}^m$ denotes the set of binary strings of length m. The set of edges, E, is defined as follows: a vertex $v = (i, \alpha)$ has an edge with vertex $u = (j, \beta)$ iff one of the following two conditions is satisfied:

1. $i = j$ and α differs from β only in the i^{th} bit.

2. $|i - j| \mod m = 1$ and $\alpha = \beta$.

From [104] we know that $b(CCC_m) = \lceil \frac{5m}{2} \rceil - 1$.

2.1.7 Butterfly Network BF_m

The m-dimensional butterfly network, BF_m, is a graph $BF_m = (V_m, E)$ where $V_m = \{0, 1, \cdots, m-1\} \times \{0,1\}^m$ and $\{0,1\}^m$ denotes the set of binary strings of length m. For any vertex $v = (i, \alpha)$ we call i the level and α the position within the level of v. The set of edges, E, is defined as follows: a vertex $v = (i, \alpha)$ has an edge with vertex $u = (j, \beta)$ iff one of the following four conditions is satisfied:

1. $i = j - 1 \mod m$ and $\alpha = \beta$.

2. $i = j + 1 \mod m$ and $\alpha = \beta$.

3. $i = j + 1 \mod m$ and α and β differ only in the j^{th} bit from the left.

4. $i = j - 1 \mod m$ and α and β differ only in the i^{th} bit from the left.

From [91] we know that $1.7417m \leq b(BF_m) \leq 2m - 1$.

2.1.8 Shuffle-Exchange Network SE_m

The SE_m is a graph on 2^m vertices where the vertices are represented by binary strings of length m. Two vertices v and u are connected iff one of the following holds:

1. v and u differ only in the last bit.

2. u is obtained from v by a single left cyclic shift.

3. u is obtained from v by a single right cyclic shift.

From [80] we know that $b(SE_m) \leq 2m - 1$.

2.1.9 de Bruijn Network DB_m

The m-dimensional de Bruijn graph [79, 11, 91], DB_m, is a graph on 2^m verices which are denoted by binary strings of length m. A vertex $v = a\alpha = \alpha' a'$, where $a, a' \in \{0, 1\}$ and α and α' are binary strings of length $m - 1$, is connected to vertices αb and $b'\alpha'$ where $b, b' \in \{0, 1\}$. From [91] we know that $b(SE_m) \geq 1.3171m$ and from [11] we know that $b(SE_m) \leq 1.5m + 1.5$.

2.1.10 2d Grid Network $G_{m,n}$

The 2 dimensional grid network $G_{m,n}$ is a network on mn vertices, having the topology of a mesh. A vertex v labeled by a the tuple (i, j) is connected to a maximum of 4 vertices, namely $(i-1, j)$, $(i, j-1)$, $(i+1, j)$, and $(i+1, j+1)$ for $1 < i < m$ and $1 < j < n$. The corner vertices have only 2 neighbours, one on each dimension. The vertices on the sides have 3 neighbours, for example $(0, j)$ is connected to $(0, j-1)$, $(0, j+1)$, $(1, j)$. From [78] we know that $b(G_{m.n}) = m + n - 2$. New results on the performance of various broadcast schemes in grids can be found in [75, 27].

2.1.11 Other Toplogies

In addition to the toplogies defined above there are other graph structures that have been studied. In [16] an optimal broadcast algorithms for star and pancake graphs is presented, and in [22] an optimal broadcast algorithm in directed graphs called the Manhattan street network is presented. In [24] broadcasting in generalized chordal rings is studied. In [69] the optimal bipartite double loop networks were considered. In [66] optimal triple loop graphs and multiloop graphs were considered and lower and upper bounds on the broadcast time were presented. In [95] a constant factor approximation algorithm is given for network of workstations. A series of papers have been devoted to the study of Knodel graphs and its broadcast time [13, 40, 41, 99, 70, 71, 54].

2.2 Approximation Algorithms and Heuristics

The broadcast problem is NP-complete so finding optimal solutions for general graph of considerable size is very inefficient. Therefore, like other NP-complete problems, the broadcast problem is sometimes addressed using approximation and heuristic algorithms.

Even though heuristic algorithms produce good results in practice, they cannot claim to have good bounds in general. Approximation algorithms are those which have a theoretical upper bound on the broadcast time in any graph. The first work of this kind is [98] where an $O(\sqrt{|V|})$ additive approximation algorithm is presented. An algorithm A is considered an $k - approximation$ scheme if the broadcast time calculated by A on a graph G, $b(G, A) \leq kb(G)$. Similarly an approximation algorithm is $k - additive$ if

$b(G, A) \leq b(G) + k$. A randomized broadcast algorithm was presented in [110] which is an $O(\frac{\log^2 |V|}{\log\log |V|})$ approximation. This algorithm is based on calculating the poise of a graph. The poise of a tree T is defined to be the maximum degree of T plus the the diameter of T. The poise of a graph G, $P(G)$ is the minimum poise of all the spanning trees of G. Calculating the poise of a graph is NP-complete. Ravi in [110] presents a $O(\log(n)P(G) + \log^2 n)$ algorithm and shows that the broadcast time of a graph G is related to the poise of G as follows: $b(G) = O(P(G)\frac{\log n}{\log\log n})$. The best theoretical upper bound is presented in [33]. Their approximation algorithm generates a broadcast algorithm with broadcast time $O(\frac{\log(|V|)}{\log\log(|V|)})b(G)$. A multicast approximation algorithm was given in [5] which is a $O(\log k)$-approximation where k is the number of recipients in the graph G. This result can trivially be generalized to broadcasting obtaining an $O(\log |V|)$ approximation.

On the other hand, [114] studies the approximability of the broadcast time and states that the broadcast time cannot be approximated within a factor of $\frac{57}{56} - \epsilon$. [32] improves the result and states that the problem is NP hard to approximate the broadcast time within a factor of $3 - \epsilon$.

The latest heuristic algorithms make use of matching-based methods. This is because each round of calls can be seen as a matching process between the sets of informed and the uninformed vertices. Extensive simulations show that the broadcasting heuristics presented in [10] and [75] have the best results. In [10] a matching based approach was utilized to derive a broadcast algorithm of time complexity $O(Rnm \log n)$, where R is the number of rounds needed to complete the broadcasting, n is the number of vertices and m is the number of edges of the graph. The heuristic in [75], TBA, reduces the complexity of each round to $O(m)$. This algorithm performs as well as the one in [10] in most of the commonly used interconnection networks and produces better results in three graph models from the network simulator ns-2 ([3, 5, 31, 119])

2.3 Construction of Broadcast and Minimum Broadcast Graphs

The previous section presented the problem of determining the minimum time needed to complete broadcasting given a graph G. Another approach in the broadcast problem is the design of graphs which optimizes the number of edges needed under the constraint that broadcasting should be completed in a certain amount of time. A graph with a broadcast time of $b(G) = \lceil \log n \rceil$ is called a broadcast graph. A minimum broadcast graph, $mbg = (V, E)$, on n vertices , $|V| = n$, is defined to be a broadcast graph with the minimum possible number of edges over all broadcast graphs on n vertices. In the literature the number of edges of an mbg on n vertices is represented by $B(n)$. There has been considerable amount of research in constructing minimum broadcast graphs [106, 100, 111, 118, 120, 116, 117, 51]. There is no known receipe for constructing graphs on n vertices for any integer value. The value of the function $B(n)$ is known only for a very few values of n. Farley et al. [37] showed that hypercubes are mbg's which implies that $B(2^m) = m2^{m-1}$. Khachatrian and Haroutunian [89] and Dineen et al. [29] independently showed that $B(2^m - 2) = (m - 1)(2^{m-1} - 1)$ for $m \geq 2$. Other than these relatively large ranges, $B(n)$ is known only for small n and for some larger values, namely $n \leq 63$ [120, 100], $n = 127$, $n = 1023$, and $n = 4095$. Direct construction of minimum broadcast graphs has proved to be a very difficult task. As a result, researchers have resorted to techniques to interconnect smaller broadcast graphs together to construct broadcast graphs on a larger number of vertices [18, 49, 19, 29]. A popular method, called the compounding technique, has proven effective for graphs on $n = n_1 n_2$ vertices since it forms the compound from two known broadcast graphs on n_1 and n_2 vertices [12, 30, 56, 89]. This approach has been quite efficient in designing graphs with even number of vertices. Another construction technique [77, 52, 14] involved the addition or deletion of a vertex in a known minimum broadcast graph.

So far we presented problems that either minimize the broadcast time in a given graph or minimize the number of edges provided that broadcasting can be completed within the theoretical minimum time. There are other optimization problems studied in the literature one of which is the problem of constructing graphs on n vertices and m edges which have the least broadcast time among all graphs on n vertices and m edges. We call this problem the (n, m) broadcast time optimization problem and (n, m) graph construction problem. To the best of our knowledge the only result in this direction is the solution of the $(n, n-1)$ problem which is the the problem of finding the minimum possible broadcast time in a tree on n vertices and the construction of such a tree. For the classical broadcast model, the results are due to Haroutunian and Khachatrian [89] and Labahn [101]. A solution for this problem in the $k - broadcasting$ model and for the universal list model can be found in [58] and [60] respectively.

Chapter 3

Unicyclic Graphs

3.1 Definitions and Auxiliary Results

A unicyclic graph (Fig. 1) is a connected graph with only one cycle. Basically it is a tree with only one extra edge. It can also be seen as a cycle where every vertex on the cycle is the root of a tree. Denote the vertices of the cycle C by $r_1, r_2, \cdots, r_k$, and the tree rooted at r_i by T_i, where $1 \leq i \leq k$. We will use the following definitions and results from [112].

Definition 1 ([112]). *The minimum broadcast time, $b_{min}(G)$, of the graph $G = (V, E)$ is defined to be the minimum of the broadcast times of all the vertices. $b_{min}(G) = min_{u \in V}\{b(u, G)\}$.*

Definition 2 ([112]). *The broadcast center of the graph G, $BC(G)$, is defined to be the set of all vertices whose broadcast time is equal to the minimum broadcast time of the graph, $BC(G) = \{u | b(u, G) = b_{min}(G), u \in V\}$.*

Theorem 1 ([112]). *Let $v \notin BC(T)$ be a vertex in a tree T such that the shortest distance from v to a vertex $x \in BC(T)$ is k. Then $b(v, T) = k + b_{min}(T)$.*

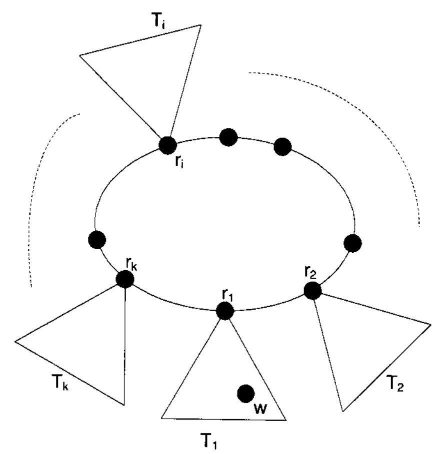

Figure 1: A unicyclic graph where the vertices r_i, belonging to the cycle C, are the roots of the trees T_i for $1 \leq i \leq k$.

Corollary 1 ([112]). *For any tree T, BC(T) consists of a star with at least two vertices.*

The unicyclic graph can be converted into a tree by cutting one of the edges of the cycle C. A simple algorithm to determine the broadcast time of a vertex w in an arbitrary unicyclic graph $G = (V, E)$ would be the SBA (SimpleBroadcastAlgorithm) algorithm provided below:

Algorithm 3.1: SBA(w, G):

1. Extract, from G, the cycle C and the trees T_i for $1 \leq i \leq k$ which are rooted at a vertex on C.

2. Cut edge (r_i, r_{i+1}) from the cycle C, for $i = 1, 2, ..., k$. Denote the resulting tree by G_i.

3. Apply BROADCAST(w, G_i) for $i = 1, 2, ..., k$ from [112] and choose the tree G_i with the minimum broadcast time $b(w, G_i)$.

The complexity of step 1 of the algorithm is $O(n)$, where $|V| = n$. The complexity of steps 2 and 3 are $O(k)$ and $O(kn)$ respectively. Thus, the total complexity of the above algorithm will be $O(kn)$, which is $O(n^2)$ in the worst case. However, $\Omega(n)$ is an obvious lower bound. In this chapter we will show $\Theta(n)$ bound by describing a linear algorithm that determines the broadcast time of any vertex w in an arbitrary unicyclic graph.

Definition 3. *Given trees $T_1 = (V_1, E_1)$, $T_2 = (V_2, E_2)$, $\cdots$, $T_i = (V_i, E_i)$ with roots r_1, r_2, $\cdots$, r_i respectively, the tree $T_{1,2,\cdots,i} = (V, E) = T_1 \oplus T_2 \oplus \cdots \oplus T_i$ is a tree where $V = V_1 \cup V_2 \cup \cdots \cup V_i$ and $E = E_1 \cup E_2 \cup \cdots \cup E_i \cup \{(r_1, r_2), (r_2, r_3), \cdots, (r_{i-1}, r_i)\}$.*

In other words, the trees T_i are connected by adding the edges (r_1, r_2), (r_2, r_3), $\cdots$, (r_{i-1}, r_i).

3.1.1 The broadcast center of the sum of two trees

In this section we will describe how to find a broadcast center and calculate the minimum broadcast time of the sum of two trees.

Lemma 1. *In any tree T, rooted at r, there exists a unique vertex $u \in BC(T)$, called the special broadcast center denoted as $u = SBC(T)$, such that the path joining u and r does not contain any other vertex v such that $v \in BC(T)$.*

Proof. The existence of vertex $u = SBC(T)$ follows from Corollary 1 and the uniqueness of $u = SBC(T)$ immediately follows from the fact that T is a tree and no cycles are allowed in a tree. □

In the remaining part of this section it is assumed that there are two trees T_1 and T_2 with roots r_1 and r_2 respectively, $T = T_1 \oplus T_2$, $u_1 = SBC(T_1)$, and $u_2 = SBC(T_2)$. We denote by $t_1(x)$ the minimum time that is needed to inform all the vertices of T_1 starting at the originator x. The vertex x does not necessarily have to be in T_1, it could be in T_2. Similarly, we denote by $t_2(x)$ the minimum time that is needed to inform all the vertices of T_2 starting at the originator x, where again x could be in T_1 or T_2.

Lemma 2. *Let x be a vertex on the path joining u_1 and u_2, then* $\max\{t_1(x), t_2(x)\} \leq b(x, T) \leq \max\{t_1(x), t_2(x)\} + 1$.

Lemma 2 can be refined to get a better understanding of the bounds on the broadcast time. For the purpose of proving the following theorems, we will present a more detailed lemma which for different conditions states the broadcast time of a vertex x, or the bounds on the broadcast time in case the exact broadcast time cannot be calculated.

Lemma 3. *The broadcast time of a vertex x which is on the path joining u_1 and u_2 satisfies the following:*

1. $b(x, T) = \max\{t_1(x), t_2(x)\}$ *if* $x \notin BC(T_1)$, $x \notin BC(T_2)$, *and* $t_1(x) \neq t_2(x)$.

2. $b(x, T) = \max\{t_1(x), t_2(x)\} + 1$ *if* $x \notin BC(T_1)$, $x \notin BC(T_2)$, *and* $t_1(x) = t_2(x)$.

3. $\max\{t_1(x), t_2(x)\} \leq b(x, T) \leq \max\{t_1(x), t_2(x)\} + 1$ *if* $x \in BC(T_1)$ *or* $x \in BC(T_2)$.

The third case of the lemma gives the bounds on the broadcast time. It states that if x is a broadcast center of T_1 or T_2 then it is possible to have $t_1(x) \neq t_2(x)$ and $b(x, T) =$

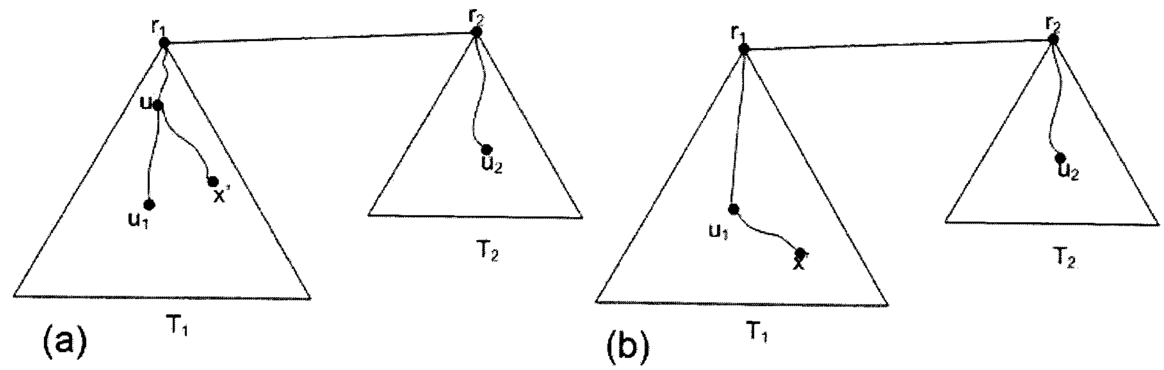

Figure 2: Sum of two trees.(a) shows the case where the hypothetical vertex x intersects the path from u_1 to r_1. (b) shows the case where the path from vertex x to r_1 goes through u_1.

$\max\{t_1(x), t_2(x)\} + 1$. The broadcast time can only be calculated if the exact tree topology is known. Later in the chapter we will present an algorithm to calculate the broadcast time of x for the case where the third condition of the lemma is satisfied.

Theorem 2. *Given two trees T_1 and T_2, there exists a vertex u such that $u \in BC(T_1 \oplus T_2)$ and u is on the path joining $u_1 = SBC(T_1)$ and $u_2 = SBC(T_2)$.*

Proof. We will prove this theorem by contradiction. Assume that there exists a vertex x' (Fig. 2) not on the path from u_1 to u_2 and such that $b(x', T) < b(u, T)$ for all vertices u on the path joining u_1 and u_2. Without loss of generality assume that x' is in T_1. Because T is a tree, there exists a unique path P that joins x' to r_1. Two cases may arise:

Case 1: The path P intersects the path from u_1 to r_1 at a vertex other than u_1 (Fig. 2a). Let $u \in P$ be this intersection vertex. Therefore, $b(u, T_1) = t_1(u) = d(u, u_1) + b_{min}(T_1)$ and $t_1(x') = b(x', T_1) = d(x', u) + (d(u, u_1) + b_{min}(T_1)) = d(x', u) + t_1(u)$. Similarly, we have $t_2(u) = d(u, r_1) + 1 + d(r_2, u_2) + b_{min}(T_2) = d(u, u_2) + b_{min}(T_2)$ and $t_2(x') =$

$d(x', u) + d(u, u_2) + b_{min}(T_2)$. Since x' does not fall on the path from u_1 to u_2, then $d(x', u) > 0$ which implies that $t_1(x') > t_1(u)$ and $t_2(x') > t_2(u)$ which implies that $\max\{t_1(x'), t_2(x')\} > \max\{t_1(u), t_2(u)\}$. Therefore, using the above lemma, we conclude that $\max\{t_1(x'), t_2(x')\} \leq b(x', T) \leq \max\{t_1(x'), t_2(x')\} + 1$ and $\max\{t_1(u), t_2(u)\} \leq b(u, T) \leq \max\{t_1(u), t_2(u)\} + 1$ which implies that $b(u, T) \leq b(x', T)$ which contradicts the assumption that $b(x', T) < b(u, T)$.

Case 2: The path P, joining x' and r_1, does not intersect the path joining u_1 and r_1. In this case the path P will merge with the path joining u_1 and r_1 at vertex u_1 (Fig. 2b). Consider $t_1(x') = d(x', u_i) + t_1(u_1)$, where $u_i \in BC(T_1)$. We are assuming that $x' \notin BC(T_1)$. The case when $x' \in BC(T_1)$ needs more careful analysis. Note that the vertex u_i can be either u_1 or at a distance of 1 or 2 from u_1 since the broadcast center of a tree is a star [112]. So we can write $t_1(x') = k + t(u_1, T_1)$ where $k \geq 1$ is the distance $d(x', u_i)$. Similarly, $t_2(x') = d(x', u_i) + d(u_i, u_2) + b_{min}(T_2)$. It can also be written as $t_2(x') = k + \delta + d(u_1, u_2) + b_{min}(T_2)$, which is equal to $t_2(x') = k + \delta + t_2(u_1)$ where $\delta = 0$ if $u_i = u_1$, $\delta = 1$ if u_i is a neighbor of u_1, or $\delta = 2$ if u_i is at distance 2 from u_1. Using the same argument as above we conclude that $b(x', T) > b(u_1, T)$ which is a contradiction.

If $x' \in BC(T_1)$ then $d(x', u_i) = 0$ so the argument used above cannot be directly applied. In this case $t_1(x') = t_1(u_1)$ and $t_2(x') = \beta + t_2(u_1)$ where $\beta = 1$ or $\beta = 2$. Therefore, we can conclude that: $\max\{t_1(x'), t_2(x')\} \geq \max\{t_1(u_1), t_2(u_1)\}$. Using the above lemma we get: $\max\{t_1(x'), t_2(x')\} \leq b(x', T) \leq \max\{t_1(x'), t_2(x')\} + 1$ and $\max\{t_1(u_1), t_2(u_1)\} \leq b(u_1, T) \leq \max\{t_1(u_1), t_2(u_1)\} + 1$. Since $b(x', T)$ can have 2 possible values and $b(u_1, T)$ too can have 2 possible values, there are 4 combinations of the pair. We need to consider each combination and show that it leads to a contradiction.

Out of the four combinations three of them can be analysed easily to find out that they lead to a contradiction. One of those situations is when $b(x', T) = \max\{t_1(x'), t_2(x')\} + 1$ and $b(u_1, T) = \max\{t_1(u_1), t_2(u_1)\} + 1$. In this case we conclude that $b(x', T) \geq b(u_1, T)$. The combination $b(x', T) = \max\{t_1(x'), t_2(x')\} + 1$ and $b(u_1, T) = \max\{t_1(u_1), t_2(u_1)\}$ leads to the conclusion that $b(x', T) > b(u_1, T)$. The third combination $b(x', T) = \max\{t_1(x'), t_2(x')\} + 1$ and $b(u_1, T) = \max\{t_1(u_1), t_2(u_1)\} + 1$ results in the inequality $b(x', T) \geq b(u_1, T)$. In all these 3 cases we see a contradiction because we assumed that there exists a vertex x' such that $b(x', T) < b(u_1, T)$.

The fourth possible combination is when $b(x', T) = \max\{t_1(x'), t_2(x')\}$ and $b(u_1, T) = \max\{t_1(u_1), t_2(u_1)\} + 1$, then using $\max\{t_1(x'), t_2(x')\} \geq \max\{t_1(u_1), t_2(u_1)\}$ we cannot conclude that $b(x', T) \geq b(u_1, T)$. Here we need to carefully analyze the broadcast scheme of x' to obtain a contradiction. Since $b(x', T) = \max\{t_1(x'), t_2(x')\}$ then we conclude that $b(x', T) = b_{min}(T) = b_{min}(T_1)$. Both vertices x' and u_1 are broadcast centers of T_1. There are 3 possibilities: x' is the center of the star that is formed by the broadcast centers of T_1, u_1 is the center of the star, or neither x' nor u_1 is the center of the star. When neither x' and u_1 is the center of the star, then if there is a broadcast scheme in T that when originating at x' gives a broacast time of $b_{min}(T_1)$ then one can construct a broadcast scheme that has a broadcast time of $b_{min}(T_1)$ when originating at u_1. If x' is the center of the star that forms the broadcast center of T_1 then again a broadcast scheme can be constructed for u_1 using the optimal broadcast scheme of x'. In both cases, when u_1 is the originator then u_1 informs the center of the star at time 1, which knows how to do broadcasting in the remaining of the tree. After the first time unit, u_1 starts informing the subtree attached to it using the same scheme that it used when the originator was x'.

The third case is when u_1 is the center of that star that forms the broadcast center of T_1, and x' is not the center. In this case, one can again modify the broadcast scheme for the originator x' to get a broadcast scheme of u_1 such that $b_{min}(u_1, T) = b(x', T) = b_{min}(T_1)$. In the broadcast scheme for the originator x', x' informs u_1 first since u_1 is the center of the broadcast centers, then u_1 forwards the message to all the other broadcast centers then informs the rest of the tree. Meanwhile, each broadcast center informs the subtree attached to it. In the broadcast scheme of u_1, the originator, u_1, informs all the broadcast centers in the order from the one the highest label to the one with the smallest label. Note that the neighbouring vertex of u_1 that is on the path from u_1 to r_2 is not a broadcast center. So the time at which it is informed is still the same whether x' or u_1 is the originator. Therefore, u_1 uses the broadcast scheme that it used when the originator was x' to inform the tree attached to it. As a result, we obtain a scheme for the originator u_1 with broadcast time $b_{min}(T_1)$. In conclusion, the case $b(x', T) = \max\{t_1(x'), t_2(x')\}$ and $b(u_1, T) = \max\{t_1(u_1), t_2(u_1)\} + 1$ always leads to a contradiction which implies that there is no case where $b(x', T) < b(u_1, T)$. □

Let u be a vertex such that $u \in BC(T_1 \oplus T_2)$. Theorem 2 confirms the existence of such a vertex on the path joining u_1 and u_2. The position of u can be found as described in the following theorem.

Theorem 3. *Let $A = b_{min}(T_1) - b_{min}(T_2)$ and $B = d(r_1, u_1) + d(r_2, u_2) + 1$. Three cases may arise:*

If $B - A < 0$, then $d(u, u_1) = 0$ and $u = u_1$, i.e. $u_1 \in BC(T_1 \oplus T_2)$.

If $A + B < 0$, then $d(u, u_1) = B$ and $u = u_2$, i.e. $u_2 \in BC(T_1 \oplus T_2)$.

If $B - A \geq 0$ and $A + B \geq 0$, then $d(u, u_1) = \lfloor \frac{B-A}{2} \rfloor$ or $d(u, u_1) = \lceil \frac{B-A}{2} \rceil$. Both positions of u have equal broadcast times in the tree T.

Proof. If $B - A < 0$, then we have:

$$b_{min}(T_1) > b_{min}(T_2) + d(u_1, r_1) + 1 + d(u_2, r_2). \tag{1}$$

We will prove that $u = u_1 \in BC(T)$ by contradiction. Assume that there is another vertex u' on the path joining u_1 and u_2 such that $b(u', T) < b(u, T)$. Because of Theorem 2 we do not have to consider a vertex not belonging to the path joining u_1 and u_2. Using the definitions of the functions $t_1(x)$ and $t_2(x)$ from above we get: $t_1(u') = d(u', u_1) + b_{min}(T_1)$, $t_2(u') = d(u', r_1) + 1 + d(u_2, r_2) + b_{min}(T_2)$, $t_1(u) = b_{min}(T_1)$, and $t_2(u) = d(u_1, r_1) + 1 + d(u_2, r_2) + b_{min}(T_2)$. Using the condition in equation 1 we get that $t_1(u) = b_{min}(T_1) > t_2(u) = d(u_1, r_1) + 1 + d(u_2, r_2) + b_{min}(T_2)$ and conclude that $t_1(u) \leq b(u, T) \leq t_1(u) + 1$. Similarly, $t_1(u') = d(u', u_1) + b_{min}(T_1) > d(u', u_1) + b_{min}(T_2) + d(u_1, r_1) + 1 + d(u_2, r_2)$ which implies that $t_1(u') > d(u', u_1) + b_{min}(T_2) + d(u_1, u') + d(u', r_1) + 1 + d(u_2, r_2) = 2d(u', u_1) + b_{min}(T_2) + d(u', r_1) + 1 + d(u_2, r_2) = 2d(u', u_1) + t_2(u')$ which implies that $t_1(u') > t_2(u')$ and hence $t_1(u') \leq b(u', T) \leq t_1(u') + 1$. On the other hand, we have $t_1(u') = d(u', u_1) + b_{min}(T_1)$ and $t_1(u) = b_{min}(T_1)$ which implies that $t_1(u') > t_1(u)$.

Therefore we conclude that $b(u', T) \geq b(u, T)$ which contradicts the assumption. The case $A + B < 0$ can be proved similarly.

Note that the two conditions $A + B < 0$ and $B - A < 0$ are mutually exclusive. If either one of them is satisfied the other will not be satisfied. The only remaining case is when both of them are not satisfied i.e. $A + B \geq 0$ and $B - A \geq 0$. Assume the case where $B - A$ is odd, the case if it is even can be dealt with similarly. Without loss of generality, assume that there exists a vertex $u' \in T_1$, on the path joining u_1 and u_2, such that $b(u', T) < b(u, T)$. Two cases may arise:

Case 1: $d(u', u_1) < d(u, u_1) = \lfloor \frac{B-A}{2} \rfloor$. Since $B - A$ is assumed to be odd, $d(u, u_1) = \frac{B-A-1}{2}$. Calculating $t_1(u) = d(u, u_1) + b_{min}(T_1)$ and $t_2(u) = d(u, r_1) + 1 + d(u_2, r_2) + b_{min}(T_2)$ we deduce that $t_2(u) = (d(u_1, r_1) - d(u, u_1)) + 1 + d(u_2, r_2) + b_{min}(T_2)$. Substituting the value of $d(u, u_1)$, and $b_{min}(T_2) = b_{min}(T_1) - A$ we get: $t_2(u) = [d(u_1, r_1) + 1 + d(u_2, r_2)] + b_{min}(T_2) - d(u, u_1) = B + (b_{min}(T_1) - A) - \frac{B-A-1}{2} = \frac{B-A+1}{2} + b_{min}(T_1) = t_1(u) + 1$. Therefore, $b(u, T) = t_2(u)$. Now consider the vertex u', $t_1(u') = d(u', u_1) + b_{min}(T_1)$ and $t_2(u') = d(u', u) + d(u, r_1) + 1 + d(u_2, r_2) + b_{min}(T_2)$. Since $d(u', u_1) < d(u, u_1)$, we get $t_1(u') < t_1(u)$. Moreover we conclude that $t_2(u') = d(u', u) + t_2(u) > t_1(u')$ since $t_2(u) > t_1(u) > t_1(u')$. Finally we get $b(u', T) = t_2(u') > b(u, T)$ if $d(u', u) > 1$ and $b(u', T) \geq b(u, T)$ if $d(u', u) = 1$, both of which contradict the assumption.

Case 2: $d(u', u_1) > d(u, u_1)$. As it was done in the previous case, we can deduce that $t_2(u) = t_1(u) + 1$ and $b(u, T) = t_2(u)$. Now consider the vertex u'. Calculating $t_1(u')$ and $t_2(u')$ we get: $t_1(u') = d(u', u) + d(u, u_1) + b_{min}(T_1)$ and $t_2(u') = d(u', r_1) +$

$1 + d(r_2, u_2) + b_{min}(T_2)$. Using that $d(u, r_1) = d(u, u') + d(u', r_1)$ we get that $t_2(u') = [d(u, r_1) - d(u, u')] + 1 + d(r_2, u_2) + b_{min}(T_2)$. Hence, $t_2(u') = t_2(u) - d(u, u') = t_1(u)+1-d(u, u')$. On the other hand, $t_1(u') = t_1(u)+d(u, u')$. Since $d(u', u_1) > d(u, u_1)$, we conclude that $d(u, u') \geq 1$. Subtracting $t_2(u')$ from $t_1(u')$ we get: $t_1(u') - t_2(u') = t_1(u) + d(u, u') - [t_1(u) + 1 - d(u, u')]$. Therefore, $t_1(u') - t_2(u') = 2d(u, u') - 1$. Using $d(u, u') \geq 1$, we get that $t_1(u') > t_2(u') + 1$. Using the values of $t_1(u')$ and $t_2(u')$ we can calculate the bounds on the broadcast time $b(u', T)$: $t_1(u') \leq b(u', T) \leq t_1(u') + 1$. Using the following: $t_2(u') = t_2(u) - d(u, u')$ $t_1(u') = t_2(u') + 2d(u, u') - 1$ We can deduce that: $t_1(u') = t_2(u) + d(u, u') - 1$. So, $t_2(u) + d(u, u') - 1 \leq b(u', T) \leq t_2(u) + d(u, u')$. Since $d(u, u') \geq 1$ and $b(u, T) = t_2(u)$ we deduce that $b(u', T) \geq b(u, T)$ which is a contradiction. Finally, we want to note that according to the theorem there are 2 vertices on the path that are broadcast centers. One of them is u and the other one is its neighbor on the path from u to u_2. □

The above theorem (Theorem 3) gives an algorithm for calculating the minimum broadcast time of the sum of two trees when the conditions $B - A \geq 0$ and $A + B \geq 0$ are satisfied. Together with Theorem 2 one can conclude that the vertices calculated by the proposed formula are the only broadcast centers of the sum of two trees. However, in the case if $B - A < 0$ or $A + B < 0$ the theorem only finds one broadcast center of the graph which is either the $SBC(T_1)$ or $SBC(T_2)$. In this case, finding the broadcast time of the sum of two trees is not straightforward. Lemma 3 states tight upper and lower bounds on the broadcast time. Before we present the broadcast algorithm for the unicyclic graphs in the next section, we need to present an algorithm to calculate the minimum broadcast time

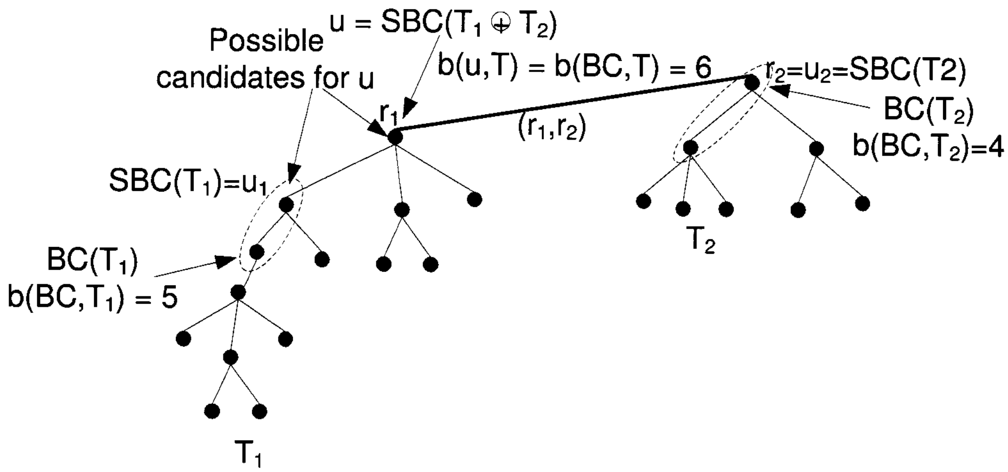

Figure 3: An example of the sum of two trees. It shows the vertices in the broadcast center and the special broadcast center of T_1, T_2, and $T_1 \oplus T_2$. The minimum broadcast time of the three trees are calculated too.

for case 3 of Lemma 3.

In the tree broadcast algorithm of [112] every time a vertex, v, is informed it informs its children in order from the one with the highest label to the one with the smallest label. Assuming that the children of v are $v_1, ..., v_k$, such that $l(v_1) \geq \cdots \geq l(v_k)$ where $l(v_i)$ denotes the label of vertex v_i, the label of the vertex v is determined by that of the children as follows: $l(v) = \max_{1,\cdots,k}\{l(v_i) + i\}$. When vertex v is informed it has to inform all the j vertices, such that $l(v) = l(v_j) + j$, during the first j time units otherwise the broadcast time of the vertex v in the subtree attached to it will not be equal to $l(v)$.

We will define two quantities, the first one which we call free time unit of a vertex v, $FTU(v)$. This is the number of time units that the vertex v should spend informing its children in the order from the highest label to smaller ones, before it can spend one idle time unit and then continue the broadcast process. From the argument in the previous paragraph one can conclude that $FTU(v) = j$ where j is the largest integer such that $l(v_j) + j = l(v)$. The second quantity is called Turn Delay Time, $TDT(v, v_i)$, which indicates the time delay after which the parent v informs its child vertex v_i in the tree broadcast algorithm of [112]. Assuming that the children of v are denoted by $v_1, \cdots, v_k$, such that $l(v_1) \geq \cdots \geq l(v_k)$, then $TDT(v, v_i) = i$.

In order to be able to calculate the broadcast time of the sum of two trees we need to calculate the quantities $FTU(v)$ and $TDT(v, v_i)$ for some of the vertices while running the tree broadcast algorithm of [112]. Given a tree T_1 rooted at r_1 , the quantities $FTU(v)$ and $TDT(v, v_i)$ will be calculated for all the vertices on the path from vertex r_1 to the closest broadcast center vertex, u_1, of T_1. In the bottom up algorithm of [112] the label of r_1 is calculated when the labels of all the children of r_1 have been calculated, in addition to that the following has to be done.

1. Calculate $FTU(r_1)$. $FTU(r_1)$ is the number of time units that r_1 should spend informing its children before it can take one time unit to inform the root of the attached tree while keeping the label of r_1 in T unchanged.

2. Calculate the distance from r_1 to r_2, the root of the tree T_2 which will be attached to T_1. Since, r_1 is the root of T_1 then the distance $d(r_1, r_2) = 1$.

For all the other vertices v_i on the path from r_1 to u_1 the following has to be calculated. Note that for convenience, the vertex r_1 will be denoted by v_1.

1. Calculate the distance from v_i to r_2, $d(v_i, r_2)$ as follows: $d(v_i, r_2) = d(v_{i-1}, r_2) + 1$.

2. The earliest time that r_2 can be informed without changing the label of v_i.

We will call this time $t(v_i, r_2)$. The child of v_i that is on the path from v_i to r_2 is v_{i-1}, therefore we need to calculate the time units that v_i should spend informing its children before it can take one time unit to inform its child v_{i-1} which is on the path to r_1. Two cases can arise:

1. If $FTU(v_i) < TDT(v_i, v_i - 1)$, then $t(v_i, r_2) = FTU(v_i) + d(v_i, r_2)$.

2. If $FTU(v_i) \geq TDT(v_i, v_{i-1})$, then $t(v_i, r_2) = TDT(v_i, v_{i-1}) + t(v_{i-1}, r_2)$.

This recursive equation can be solved by noting that the base case is $t(v_1, r_2) = FTU(r_1) + 1$ where $r_1 = v_1$.

Given a tree T_1 rooted at r_1 which is connected to root r_2 of another tree T_2 we showed how to calculate the quantity $t(SBC(T_1), r_2)$. Now using this result we will show how to calculate in constant time the minimum broadcast time of the sum of the trees T_1 and T_2 for the cases where $B - A < 0$ or $A + B < 0$. We will consider the case where $B - A < 0$, the other case $A + B < 0$ is very similar to the first one. $B - A < 0$ implies that $b_{min}(T_1) > b_{min}(T_2) + d(r_1, u_1) + d(r_2, u_2) + 1$. Assume that u_0 is the center of the star that forms the broadcast centers of T_1. If $SBC(T_1) = u_0$, then two cases arise:

1. $t(u_0, r_2) + d(r_2, u_2) + b_{min}(T_2) \leq b_{min}(T_1)$ then we can conclude that $b_{min}(T) = b_{min}(T_1)$.

2. $t(u_0, r_2) + d(r_2, u_2) + b_{min}(T_2) > b_{min}(T_1)$ then we can conclude that $b_{min}(T) = b_{min}(T_1) + 1$.

If $u_1 = SBC(T_1)$ is not the center of the star (i.e. it is a neighbor of u_0), then we need to first consider the relation between the label of u_0 and u_1. If $l(u_1) < l(u_0) - 1$, then $b_{min}(T) = b_{min}(T_1)$. If $l(v_1) = l(u_0) - 1$ then the following cases should be considered:

1. $t(u_1, r_2) + d(r_2, u_2) + b_{min}(T_2) \leq b_{min}(T_1) - 1$ then we can conclude that $b_{min}(T) = b_{min}(T_1)$.

2. $t(u_1, r_2) + d(r_2, u_2) + b_{min}(T_2) > b_{min}(T_1) - 1$ then we can conclude that $b_{min}(T) = b_{min}(T_1) + 1$.

Theorem 4. *For $T = T_1 \oplus T_2$, if $b_{min}(T) = b_{min}(T_1)$ then $BC(T_1) = BC(T)$.*

In what follows we will denote the neighbor of u_1 that is on the path from u_1 to r_1 by u_1'. It is necessary to calculate the label of u_1' when T_2 is attached to r_1. The only scenario we are interested in is when $b_{min}(T) = b_{min}(T_1) + 1$ because this is the only condition for which the broadcast centers of T_1 can be different that those of T.

Lemma 4. *If $b_{min}(T) = b_{min}(T_1) + 1$ then $l(u_1', T) = \max\{l(u_1', T_1) + 1, d(u_1', r_2) + l(r_2, T_2)\}$ where $l(v, T)$ denotes the label of a vertex v in the tree T.*

The justification for this is due to the fact that the label of u_1' must be increased by at least 1 otherwise the broadcast time of u_1 cannot increase by 1. Moreover, the label of u_1' should be large enough so that the vertex r_2 can be informed and can have enough time to inform T_2. See Figure 3 for an example.

The labels of the neigbhors of u_1 remain unchanged when T_2 is added except that of u_1'. Let the neigbhors of u_1 be denoted by $w_1, \cdots, w_k$ and labeled such that $l(w_1, T) \geq \cdots \geq l(w_k, T)$.

Theorem 5. *If $b_{min}(T) = b_{min}(T_1) + 1$ then the center of the star that forms the broadcast center of T is $u_1 = SBC(T_1)$. The other broadcast centers are those neighbors w_i, $1 \leq i \leq p$, such that $l(w_p, T) + p = b_{min}(T)$ for the smallest possible value of p.*

3.2 The Unicyclic Graph Broadcast Algorithm

The algorithm, $UBA(w, G)$ (UnicyclicBroadcastAlgorithm), calculates the broadcast time, $b(w, G)$, of a given vertex w in any unicyclic graph G. For convenience we will assume that w belongs to tree T_1. We denote the shortest distance from a vertex v to a vertex belonging to the broadcast center of T by $d(v, BC(T))$.

3.2.1 Description of the Algorithm

INPUT: A unicyclic graph G on n vertices and the broadcast originator w.

OUTPUT: Broadcast time of the originator w in G, $b(w, G)$, and a broadcast scheme.

Algorithm 3.2: UBA(w, G):

1. Extract from G the cycle C, consisting of the vertices $\{r_1, r_2, \cdots, r_k\}$, and the trees T_i rooted at the vertices r_i for $1 \leq i \leq k$.

2. For all trees T_i where $1 \leq i \leq k$ calculate and save the positions of $u_i = SBC(T_i)$ relative to r_i, $b_{min}(T_i)$, as well as $t(v_i, r_{i+1})$ for every vertex v_i on the path from r_i to $SBC(T_i)$.

3. Calculate and save the distance $d(w, BC(T_1))$ and the path joining w and u_1.

4. Construct the trees $T_{1,2,\cdots,i}$, where $2 \leq i \leq k$, and $T_{k,k-1,\cdots,i}$, where $1 \leq i \leq k-1$. For each tree T, compute and store $BC(T)$ and $b_{min}(T)$.

5. Construct the spanning trees $T_{j,j+1,\cdots,k,1,2,\cdots,j-1}$, where $1 \leq j \leq k$. For each tree T compute and store $BC(T)$, $b_{min}(T)$, $d(w, BC(T))$, and $b(w, T)$.

6. Out of the trees generated in the previous step, choose the spanning tree T with the minimum value of $b(w, T)$.

7. Run BROADCAST [112] to find a broadcast scheme for the originator w.

The algorithm first preprocesses the unicyclic graph and calculates the cycle C consisting of the vertices $\{r_1, r_2, \cdots, r_k\}$ and the trees T_i rooted at r_i , where $1 \leq i \leq k$. In step 3 the path joining w to u_1 and $d(w, BC(T_1))$ are calculated and saved. This information will be needed to calculate the broadcast time of w. Steps 4 and 5 construct a set of trees and calculate results that will be used in constructing the k spanning trees of the graph G. More specifically the trees, T_i, $T_{i,i+1,\cdots,k}$ for $1 \leq i \leq k-1$, and $T_{1,2,\cdots,i}$ for $2 \leq i \leq k$ are constructed. For each tree T the $BC(T)$ and $b_{min}(T)$ are calculated and stored. These results will be useful to calculate the broadcast centers and the minimum broadcast times of the spanning trees. At the end of step 4 only one spanning tree will be constructed which is $T_{1,2,\cdots,k}$. In step 5, the algorithm builds the remaining $k-1$ spanning trees of the unicyclic graph. It also calculates the broadcast time of w for each one of them. Note that for each spanning tree T, $b(w, T) = d(w, BC(T)) + b_{min}(T)$ where the distance $d(w, BC(T)$ can be easily calculated by using $d(w, u_1)$ and position of $SBC(T)$ calculated in steps 2 and 4 respectively. The spanning tree that has the minimum broadcast time for w is the required result. Finally in order to obtain the optimal broadcast scheme for the originator w in the unicyclic graph G, the BROADCAST algorithm of [112] is run on the spanning tree T that had the minimum value of $b(w,T)$.

In step 5, $k-1$ spanning trees are constructed each in constant time. The construction of each tree can be done easily by observing that the spanning tree $T_{i,i+1,\cdots,k,1,2,\cdots,i-1}$ is the sum of the two trees $T_{i,i+1,\cdots,k}$ and $T_{1,2,\cdots,i-1}$ rooted at r_k and r_1 respectively. These two trees and all the information pertinent to the calculation of the $b(w,T_{i,i+1,\cdots,k,1,2,\cdots,i-1})$ were calculated in step 4.

3.2.2 Proof of Correctness and Complexity Analysis

Theorem 6. *For a unicyclic graph G, $UBA(w,G)$ generates a broadcast scheme for the originator w, and calculates $b(w,G)$.*

Proof. Let C, consisting of the vertices $\{r_1, r_2, \cdots, r_k\}$, represent the cycle in the unicyclic graph G. Removing the edge (r_k, r_1) results in a spanning tree of G which is $T_{1,2,\cdots,k}$. The minimum broadcast time of $T_{1,2,\cdots,k}$ is calculated iteratively by tree summations $T_{1,\cdots,i-1} \oplus T_i$ rooted at r_{i-1} and r_i respectively, where $2 \leq i \leq k$. The remaining minimum spanning trees, $T_{j,j+1,\cdots,k,1,2,\cdots,j-1}$ where $2 \leq j \leq k$, are calculated by performing the summations $T_{j,\cdots,k} \oplus T_{1,\cdots,j-1}$. Theorems 2 and 3 guarantee that a broadcast center and the minimum broadcast time of all the trees $T_{j,j+1,\cdots,k,1,2,\cdots,j-1}$, where $1 \leq j \leq k$, are calculated correctly. Since, the trees $T_{j,j+1,\cdots,k,1,2,\cdots,j-1}$, where $1 \leq j \leq k$, are the all possible spanning trees of the unicyclic graph G, the algorithm correctly finds the spanning tree of the unicyclic graph G that has the minimum broadcast time of all the spanning trees. We will prove the correctness of the algorithm by contradiction. Assume that there exists a broadcast tree T' and a broadcast scheme in G that performs broadcasting in time t such that $t = d(w, BC(T')) + b_{min}(T') < b(w, G)$. T' should be one of the trees $T_{j,j+1,\cdots,k,1,2,\cdots,j-1}$, where $1 \leq j \leq k$. But the algorithm correctly calculated the minimum broadcast time of all the spanning trees and chose the spanning tree T with the minimum value of $d(w, BC(T)) + b_{min}(T)$. Therefore the existence of T' creates a contradiction. □

Theorem 7. *The complexity of the UBA algorithm running on a unicyclic graph* $G = (V, E)$ *is* $O(|V|)$.

Proof. Steps 1 and 2 of the algorithm can be accomplished by a depth first search in $O(|V|)$ time. In Step 3, BROADCAST [112] is applied on the trees T_i for $1 \leq i \leq k$. The complexity of this step is $O(|V_1| + |V_2| + \cdots + |V_k|) = O(|V|)$. Step 4 is of complexity $O(k)$ since there are k sums to be done, and the sum of two trees T_1 and T_2, $T_1 \oplus T_2$, can be done in constant time. Moreover, every time a tree T is constructed as $T = T_1 \oplus T_2$, calculating the distance between the root of T and $SBC(T)$, and $b_{min}(T)$ can be done in constant time using the positions of $SBC(T_1)$ and $SBC(T_2)$, and the broadcast times $b_{min}(T_1)$ and $b_{min}(T_2)$. Step 5 involves the calculation of $k - 1$ spanning trees. Each spanning tree is constructed by summing two trees, hence the complexity of this step is again $O(k)$. Step 6 chooses the spanning tree with the least minimum broadcast time hence its complexity is $O(k)$. Adding all the complexities we get that the complexity of the algorithm is $O(|V| + k)$. However, $k \leq |V|$, so this proves that the complexity of the algorithm UBA is $O(|V|)$. □

3.3 The Broadcast Time of a Unicyclic Graph

So far, we showed how to calculate the minimum broadcast time of a unicyclic graph and how to caluclate the broadcast time of a vertex in a unicyclic graph. Another problem which is usually studied in the literature is finding $b(G)$, the broadcast time of the given graph G. There is an obvious algorithm that will run in $O(kn)$ time. In this section we will present an algorithm to calculate the broadcast time of an arbitrary unicyclic graph in $O(n+k^2)$ time. This is always better than the $O(kn)$ algorithm except when $k = \Theta(|V|)$. In particular, the time complexity of our algorithm is a linear function of $|V|$ when $k = O(\sqrt{(|V|)}$.

For a vertex v in a graph G let us assume that $u \in BC(G)$ is the closest vertex to v. In general, the broadcast tree of the optimum broadcast scheme of vertex v is not the same as the broadcast tree of the optimum broadcast scheme of u. Hence, calculating the broadcast center of the graph and then finding the furthest vertex from the broadcast center does not guarantee that the sum of that distance and the minimum broadcast time is the broadcast time of the graph G.

The major modification of the algorithm will involve calculating the broadcast time of each spanning tree $T_{i,\cdots,k,1,\cdots,i-1}$. The modified UBA algorithm, MUBA (Modified Unicyclic Broadcast Algorithm), is presented below:

Algorithm 3.3: MUBA(G):

1. Extract from G the cycle C, consisting of the vertices $\{r_1, r_2, \cdots, r_k\}$, and the trees

T_i rooted at the vertices r_i for $1 \leq i \leq k$.

2. For all trees T_i where $1 \leq i \leq k$ calculate save $BC(T_i)$, as well as $b_{min}(T_i)$. Also find the vertex v_i such the $d(r_i, v_i)$ is maximum.

3. Construct the trees $T_{1,2,\cdots,i}$, where $2 \leq i \leq k$, and $T_{k,k-1,\cdots,i}$, where $1 \leq i \leq k-1$. For each tree T, compute and store $BC(T)$ and $b_{min}(T)$.

4. Construct the spanning trees $T_{j,j+1,\cdots,k,1,2,\cdots,j-1}$, where $1 \leq j \leq k$. For each tree T compute and store: $BC(T)$, $b_{min}(T)$, and the broadcast time of $b(T)$

5. Out of the trees generated in the previous step, choose the spanning tree T with the minimum value of $b(T)$.

Step 4 of the above algorithm is the main place where the major modification was done. The new quantity that is being calculated is the broadcast time, $b(T)$, of each spanning tree T. This calcluation can be done in $O(k)$ time as follows. Assume that the $SBC(T)$ is in tree T_n. Calculate the quantities $d(SBC(T), r_n) + d(r_n, r_j) + d(r_j, v_j)$ for $1 \leq j \leq k$ and $i \neq j$. Also calculate the distance $d(v, BC(T))$ such that $v \in T_n$. Choose the maximum of these values, d_{max}. This is the furthest distance from $BC(T)$, hence, $b(T) = d_{max} + b_{min}(T)$.

Theorem 8. *Given a unicyclic graph $G = (V, E)$, the algorithm MUBA(G) calculates $b(G)$ in $O(|V| + k^2)$ time.*

Proof. Step 1 of the algorithm needs $O(|V|)$ time. Steps 2 and 3 take $O(k)$ time as it was argued previously. The difference from the previous algorithm lies in step 4 which is not a constant time operation anymore. For each spanning tree T, calculating $b(T)$ involves calculating k distances and finding the minimum. This needs $O(k)$ steps. Hence, the complexity of this step is $O(k^2)$. Step 5 has complexity $O(k)$. Therefore, we conclude that the complexity of the algorithm is $O(n + k^2)$. □

3.4 Lower and Upper Bounds on the Broadcast Time of Unicyclic Graph

A very important problem in this area is to find the relationship between the broadcast times of a connected graph $G = (V, E)$ and one of its connected spanning subgraphs $H = (V, E')$. Such a result is very useful for designing good approximation algorithms to determine the broadcast time of an arbitrary vertex in an arbitrary graph. This will also be used for finding or evaluating the broadcast time of a given graph when the broadcast time of its subgraph is known. As the first step in this direction we will give tight lower and upper bounds on $b(H)$ in terms of $b(G)$, where G is an arbitrary unicyclic graph.

3.4.1 Lower and Upper Bounds on the Minimum Broadcast Time

The result obtained in this section will be used in Section 5.2 to give tight lower and upper bounds on the broadcast time of a spanning tree of a unicyclic graph $G = (V, E)$. In this section we will study how much the minimum broadcast time (the broadcast time of the broadcast center) of a unicyclic graph varies when different edges of the cycle are removed (one at a time). First it is obvious that if the removed edge is not from the cycle then we will end up with two disconnected components one of which is a tree and the other one is a unicyclic graph. Therefore, we will consider only the case where the removed edge was part of the cycle. Our interest is to see how high the minimum broadcast time of a spanning tree of a unicyclic graph can be compared to the minimum broadcast time of the unicyclic graph.

It is clear that deleting one or more edges from a graph will increase the minimum broadcast time of the initial graph (as well as the broadcast time). The bounds that we are looking at do not necessarily happen in all unicyclic graphs. For example when the trees attached to the cycle are isomorphic then it will not make any difference whichever edge was removed. We will give an upper bound on the minimum broadcast time of the worst spanning tree of the unicyclic graph.

Let $G = (V, E)$ be a unicyclic graph. The k vertices of the cycle are denoted by r_i where $1 \leq i \leq k$. Without loss of generality, assume that removing the edge (r_1, r_k) yields the tree $T = T_{1,\cdots,k}$ with the minimum broadcast time t_{min}. In other words $b_{min}(G) = b_{min}(T) = t_{min}$. Now assume that the edge (r_1, r_k) is restored and another edge (r_i, r_{i+1}) is removed. The broadcast time of the resulting tree can be calculated by noticing that the tree $T' = T_{i+1,i+1,\cdots,k,1,2,\cdots,i}$ is the sum of the two trees $T_A = T_{i+1,\cdots,k}$ and $T_B = T_{1,\cdots,i}$. One important factor that decides the broadcast time of T' is the distance between the special broadcast centers of T_1 and T_2.

We use Theorem 3 to calculate the broadcast time of $T_{1,\cdots,k}$, $b(T_{1,\cdots,k}) = t_{min}$ by using the two trees $T_A = T_{1,\cdots,i}$ and $T_B = T_{i+1,\cdots,k}$.

To apply Theorem 3, we need to consider that the tree T_A is rooted at r_i and T_B is rooted at r_{i+1}. Assume that $SBC(T_A)$ is in the tree rooted at r_x and $SBC(T_B)$ is in the tree rooted at r_y. Let t_1 and t_2 be the broadcast times of $SBC(T_A)$ in T_A and $SBC(T_B)$ in T_B respectively. Let d_x be the distance between $SBC(T_A)$ and r_x and d_y be the distance between $SBC(T_B)$ and r_y. We directly apply Theorem 3 to find the broadcast centers of T. Direct substitution gives:

$A = b_{min}(T_A) - b_{min}(T_B)$ and $B = d_x + d_y + y - x$.

And the two vertices, u and u', of the broadcast center of T are at the following distance from $SBC(T_A)$:

$\lfloor \frac{B-A}{2} \rfloor$ and $\lceil \frac{B-A}{2} \rceil$.

Note that Theorem 3 has three different cases, however the first two cases lead to the situation when the new broadcast center coincides with one of the broadcast centers of the initial trees. It is easy to verify (the details are omitted) that these two cases do not lead to the upper bound that we are looking for. So, we assume that case 3 of Theorem 3 holds.

Now we calculate the broadcast time of $T' = T_{i,i+1,\cdots,k,1,2,\cdots,i}$ using the same trees T_A and T_B and the already known information about their broadcast centers. This will allow us to compare $b_{min}(T)$ and $b_{min}(T')$ because they will both be expressed in terms of the same variables. Using Theorem 3 we can find the vertices of the broadcast center. Here, we need to assume that T_A is rooted at vertex r_1 and T_B is rooted at vertex r_k. However, this does not change the previous assumption that $SBC(T_A)$ is in T_x and $SBC(T_B)$ is in T_y. Direct application of Theorem 3 gives:

$A' = b_{min}(T_A) - b_{min}(T_B)$ and $B' = d_x + d_y + k - (y - x)$.

The two possible vertices, u and u', of the broadcast center of T' are at the following distance from $SBC(T_A)$:

$\lfloor \frac{B'-A'}{2} \rfloor$ and $\lceil \frac{B'-A'}{2} \rceil$.

After finding the vertices of the broadcast center one can easily calculate the minimum broadcast time. There are two cases that one has to consider. One case is when $B-A$ is odd and the other when it is even. In the first case there will be two vertices in the broadcast center and in the latter case there will be one vertex only.

$t_{min} = b_{min}(T) = \frac{B-A}{2} + t_1 + 1$ when $B-A$ is even.

$t_{min} = b_{min}(T) = \lceil \frac{B-A}{2} \rceil + t_1$ when $B-A$ is odd.

Rewriting the two equations we get:

$t_{min} = b_{min}(T) = \frac{t_1+t_2+d_x+d_y+y-x+2}{2}$ when $B-A$ is even.

$t_{min} = b_{min}(T) = \frac{t_1+t_2+d_x+d_y+y-x+1}{2}$ when $B-A$ is odd.

Similarly the broadcast time of T' can be obtained:

$b_{min}(T') = \frac{t_1+t_2+d_x+d_y+k-(y-x)+2}{2}$ when $B'-A'$ is even.

$b_{min}(T') = \frac{t_1+t_2+d_x+d_y+k-(y-x)+1}{2}$ when $B'-A'$ is odd.

We have 4 combinations of the parity of $B-A$ and $B'-A'$. To calculate the upper bound on minimum broadcast time each combination has to be treated separately. Assume the case where $B-A$ is odd and $B'-A'$ is even. All other cases can be handled similarly. In this case we get the following: $b_{min}(T') = \frac{2t_{min}+2(x-y)+k+1}{2}$.

The broadcast time depends on two variables: $(x-y)$ and k. To maximize $b_{min}(T')$, the minimum broadcast time of T' we need to maximize $2(x-y)+k$. By definition of x and y it follows that $(x-y) \leq -1$. An upper bound on k can be obtained by considering the cycle alone. We know that the broadcast time of a cycle of size k is $\lceil \frac{k}{2} \rceil$, hence $t_{min} \geq \lceil \frac{k}{2} \rceil$. In other words, $k \leq 2t_{min}$ when k is even and $k \leq 2t_{min} - 1$ when k is odd. We can substitute these values in the equation $b_{min}(T') = \frac{2t_{min}+2(x-y)+k+1}{2}$ and show that it indeed reduces to $b_{min}(T) \leq 2b_{min}(T) - 1$. Note that $B - A$ is odd and $B' - A'$ is even which implies that k must be odd. Therefore, $t_{min} \geq \lceil \frac{k}{2} \rceil$ implies that $t_{min} \geq \frac{k+1}{2}$. Also,using the fact that $x - y \leq -1$, implies that $b_{min}(T') \leq 2b_{min}(T) - 1$.

Since the equality $b_{min}(T') = 2b_{min}(T) - 1$ is obtained only when $x - y = -1$ and $t_{min} = \lceil \frac{k}{2} \rceil$ we will study this scenario in more details and will show that if $b_{min}(G) = \lceil \frac{k}{2} \rceil$ and $x - y = -1$ we obtain that $b_{min}(T) \leq 2b_{min}(G) - 2$.

Since $x - y = -1$ we conclude that the broadcast centers of T_A and T_B fall in two neighboring trees T_i and T_{i+1}. Moreover, since the length of the cycle is k and its broadcast time is $\lceil \frac{k}{2} \rceil$, we conclude that r_i and r_{i+1} are broadcast centers of T. Note that the largest value of $b(r_j, T_j)$ can be equal to $b_{min}(T) - 2$ and this could be for the trees rooted at r_i and r_{i+1}. This can be understood by analyzing the broadcast scheme of r_i. In the first time unit it has to inform the other broadcast center. In the second time unit, both broadcast centers have to inform their neighbors on the cycle. Only after the second time unit the roots r_i and r_{i+1} will have time to inform vertices in the trees T_i and T_{i+1}. Therefore, one can calculate the worst possible value of T' in such a graph as follows: $b_{min}(T') \leq \lceil \frac{k}{2} \rceil + \max_{1 \leq j \leq k}\{b(r_j, T_j)\}$, which is equal to $b_{min}(T) \leq 2b_{min}(T) - 2$.

Therefore we can realize that upper bound $b_{min}(T') \leq 2b_{min}(T) - 1$ is not tight and the tight upper bound is $b_{min}(T') \leq 2b_{min}(T) - 2$.

Therefore, in the case of $B - A$ is odd and $B' - A'$ is even, the maximum value of $b_{min}(T')$ can be $b_{min}(T') \leq 2b_{min}(T) - 2$. It should be noted that the remaining combinations of the parities of $B - A$ and $B' - A'$ either result in broadcast times less than or equal the upper bound calculated above.

Actually we proved the following result:

Theorem 9. *If G is a unicyclic graph and T' is a spanning tree of G then* $b_{min}(G) \leq b_{min}(T') \leq 2b_{min}(G) - 2$.

In Figure 4 we present an example of a unicyclic graph where $b_{min}(T') \leq 2t_{min} - 2$. Figure 4c presents T', where $b_{min}(T') = \max\{b_{min}(P)|P$ is a spanning tree of unicyclic graph $G\} = 8$. Figure 4b shows the spanning tree T for which $b_{min}(G) = b_{min}(T) = 5$. Figure 4 presents graphs on n vertices for which $b_{min}(T') = 2b_{min}(G) - 2$, for any $n = 0($ mod $4)$.

3.4.2 Lower and Upper Bounds on Broadcast Time

In this section we will give tight lower and upper bounds on the broadcast time of a vertex in a spanning tree $T' = (V, E')$ of a unicyclic graph $G = (V, E)$ expressed in terms of $b(v, G)$, the broadcast time of v in the unicyclic graph G.

Theorem 10. *If G is a unicyclic graph and T is a spanning tree of G then* $b(G) \leq b(T) \leq 2b(G) - 1$.

Proof. To prove the lower bound consider unicyclic graph G that contains a cycle of length 3 and a path of length $n - 3$ starting from one of the vertices of the cycle. It is clear that broadcasting from the leaf of the path (the vertext on the path which is the furthest from the cycle) takes $n - 1$ time units, so $b(G) = n - 1$. It is obvious that $b(T) = b(G) = n - 1$ for any spanning tree of G.

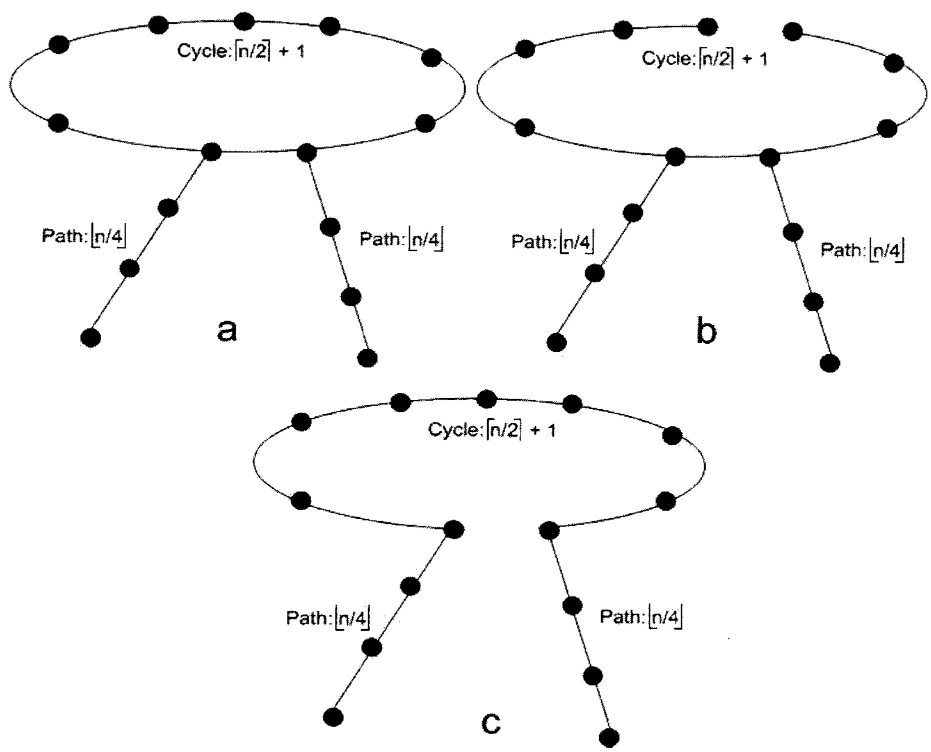

Figure 4: (a) Shows the unicyclic graph (on n vertices) that can have the maximum variation in the minimum broadcast time depending on which edge gets removed. (b) Shows the graph with the best edge removed and whose minimum broadcast time is $\lceil \frac{n}{4} \rceil + 1$ and is the minimum value possible. (c) Shows the graph with the wrong edge removed where the minimum broadcast time increases to $\lceil \frac{n}{2} \rceil$.

As for the upper bound we need to prove that $b(T) \leq 2b(G) - 1$. There exists a vertex v such that $b(v, T) = b(T)$, therefore we need to prove that $b(v, T) \leq 2b(v, G) - 1$ which will immediately imply the required result. Moreover, there exists a tree T' such that $b(v, G) = b(v, T')$, therefore, we need to prove that $b(v, T) \leq b(v, T') - 1$. As before we can assume that T is the sum of two trees T_A and T_B. Without repeating the details of the calculation as was done in the previous theorem, we can calculate the distance of the broadcast centers of T and T' from $SBC(T_A)$. What interests us is the difference in the distance which is $k + 2(x - y)$. On the other hand, we can calculate $b_{min}(T)$ in terms of $b_{min}(T')$. Again depending on the parities of $B - A$ and $B' - A'$ we get several combinations. When $B - A$ is odd while $B' - A'$ is even we get that $b_{min}(T') = b_{min}(T) + (x - y) + \frac{k-1}{2}$.

We can write that $b(v, T) = b_{min}(T) + d(v, BC(T))$ which implies that $b(v, T) = b_{min}(T') - (x - y) - \frac{k-1}{2} + d(v, BC(T))$. Using that $|d(v, BC(T')) - d(v, BC(T))| = k + 2(x - y)$ we obtain $b(v, T) \leq b_{min}(T') - (x - y) - \frac{k-1}{2} + d(v, BC(T')) + k + 2(x - y)$ which implies that $b(v, T) = b(v, T') + (x - y) + \frac{k+1}{2}$. Again we can use that $x - y \leq -1$ and $\frac{k+1}{2} \leq b(v, T')$ to obtain that $b(v, T) \leq 2b(v, T') - 1$. Since T' is the broadcast tree which has the property $b(v, T') = b(v, G)$ we can write $b(v, T) \leq 2b(v, G) - 1$. Finally since $b(v, G) \leq b(G)$ for any vertex v we can conclude that $b(T) \leq 2b(G) - 1$. □

The upper bound from this theorem is also attainable. An example of that is C_n, the cycle on n vertices. It is easy to see that the broadcast time of any spanning tree of C_n is P_n the path on n vertices and $b(P_n) = n - 1 = 2b(C_n) - 1$ when n is even.

Chapter 4

Subclasses of the Cactus Graph

Cactus graphs are defined to be connected graphs where no two cycles have more than one vertex in common. In this chapter we will study three kinds of graphs that are subclasses of the cactus graph. These are the tree of cycles, necklace graphs, and the 2-restricted cactus graphs. The tree of cycles is a connected graph where no two cycles are allowed to have any vertex in common. The necklace graph consists of cycles such that a cycle can have no more than two vertices in common with two other distinct cycles and no more than two cycles can have a vertex in common. The third graph is the 2-restricted cactus graphs where we add one more restriction to the definition of cactus graph. A 2-restricted cactus graph is a cactus graph where no more than two cycles can have more than one vertex in common. For the first two graph families we will present two linear time algorithms to find the broadcast time of an arbitrary vertex. However, we will present partial solution for the 2-restricted cactus graphs and discuss the difficulties which should be overcome in order to get a complete solution.

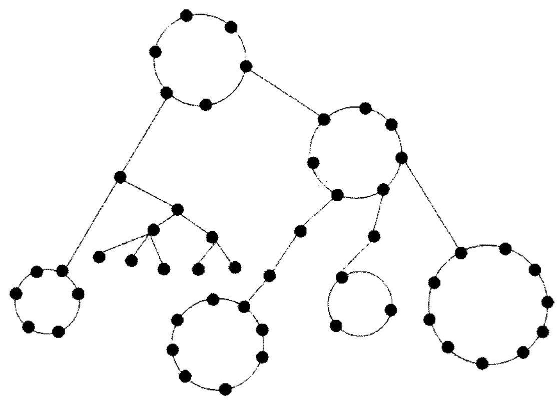

Figure 5: A tree of cycles with 6 cycles of different sizes.

4.1 Tree of Cycles

We will define a family of graphs where no two cycles intersect [65]. This can happen if and only if there is only one unique path between any two cycles. If the cycles were made up of single vertices, then this would be the definition of a tree. Hence, we can visualize this graph as a tree of cycles.

Definition 4. *A graph $TC = (V, E)$ is a tree of cycles if for every edge $e \in E$, e belongs to a simple cycle or e is a bridge. A vertex e is a bridge if its removal results in an increase in the number of connected components of the graph.*

In this section we will describe a broadcast algorithm for an arbitrary graph in the family of tree of cycles. The strategy is to divide the problem into smaller problems, solve them individually, and then join the results. We will define a recursive algorithm that can acomplish this. For a tree of cycles $TC = (V, E)$, where V is the set of vertices and E is the set of edges, we will define the subsets $V_{C_i} \subseteq V$ and $E_{C_i} \subseteq E$ as follows. For a cycle C_i in TC let the set V_{C_i} be the subset of the vertices of the cycle C_i such that every vertex

$v \in V_{C_i}$ is the endvertex of a bridge. The set E_{C_i} is the set of edges such that one of the end vertices is in V_{C_i}. In other word E_{C_i} is the set of bridges that have an end vertex on the cycle C_i.

The broadcast algorithm for an originator v_o in a tree of cycles TC starts with the cycle C_o that contains the originator v_o, and calculates the sets $V_{C_o} = \{v_1, \cdots, v_k\}$ and $E_{C_o} = \{(v_i, u_j)\}$, where $1 \leq i \leq k$, $1 \leq j \leq p$ and $p \geq k$. The vertices u_j , $1 \leq j \leq p$, are those vertices which do not belong to the cycle C_o but have neighbors which belong to V_{C_o}. The size of E_{C_o} could be greater than the size of V_{C_o} because it is possible to have a vertex in V_{C_o} connected to more than one vertex, each in a different cycle. After calculating these sets, the graph can be divided into disconnected subgraphs by removing the edges of the set E_{C_o} from the graph TC. When the edge $(v_i, u_j) \in E_{C_o}$ is removed the graph that contains u_j will be denoted by TC_j. So removing all the edges of E_{C_o} will result in p subgraphs TC_j, $1 \leq i \leq p$. Note that each one of these graphs also belongs to the family of tree of cycles. The recursive step of the algorithm is to solve the broadcast problem in the p subgraphs TC_j, $1 \leq j \leq p$, where the originator is u_j. After solving the broadcast problem in TC_j we will have a broadcast tree, T_{TC_j}, rooted at the originator u_j. The joining step of the recursive algorithm is to put together the solutions of the subgraphs TC_j and obtain a solution for the graph TC. This is done by constructing a graph $TC' = (V', E')$, which is a subgraph of TC, consisting of the cycle C_o, the trees T_{TC_j} $1 \leq j \leq p$, and the edges E_{C_o}. Note that the graph TC' is a unicyclic graph. Therefore, the joining step is to solve the broadcast problem in the unicyclic graph TC' where the originator is v_o.

In every recursive algorithm there should be a base case at which point the problem is not subdivided into smaller problems. The base case for the algorithm suggested above is when we consider a cycle C_j such that E_{C_j} is empty. In this case the solution of the broadcast problem is very simple since the broadcast time of a cycle on n vertices is $\lceil \frac{n}{2} \rceil$ and the broadcast tree is the tree resulting after removing the $\lceil \frac{n}{2} \rceil$-th edge from the originator.

Algorithm 4.1: The broadcast algorithm that calculates the broadcast time $b(v_o, TC)$ and scheme for any originator v_o in any tree of cycles TC.

Algorithm: Broadcast Algorithm for Tree of Cycles: $BroadcastTC(TC, v_o)$
Input: $TC = (V, E)$ and originator v_o
Output: Broadcast time $b(v_o, TC)$ and the broadcast tree of TC
Find the cycle C_o such that the originator v_o belongs to it ;
Calculate the sets V_{C_o} and E_{C_o} ;
if E_{C_o} *is empty* **then**
  len ← number of vertices on the cycle C_o ;
  cut the $\lfloor \frac{len}{2} \rfloor$-th edge from v_o to obtain the broadcast tree of the cycle C_o ;
  return
else
  $TC' \leftarrow C_o$;
  foreach $(v_i, u_j) \in E_{C_o}$ **do**
    remove edge (v_i, u_j) and obtain the connected subgraph TC_j ;
    call $BroadcastTC(TC_j, u_j)$ to obtain the broadcast tree T_{TC_j} ;
    Add the edge (v_i, u_j) to the graph TC' ;
    Add T_{TC_j} to TC' ;
  end
  Calculate the broadcast time and the broadcast scheme of the unicyclic graph TC' ;
end

4.1.1 Complexity Analysis and Proof of Correctness

In this section we will present the complexity analysis of the broadcast algorithm of tree of cycles, we will also prove its correctness. We first calculate the maximum number of edges that a tree of cycles on n vertices can have. Note that cycles cannot be of size 2 since they are equivalent to two cycles of size 1 connected by an edge. Let p be the number of

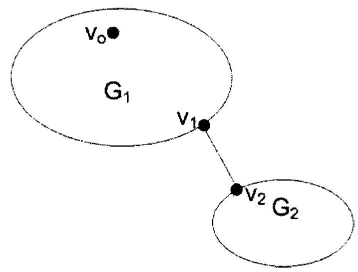

Figure 6: A graph with two connected components G_1 and G_2 connnected by the bridge (v_1, v_2).

cycles of size 1, and q be the number of cycles whose size is greater than or equal to 3. First notice that one needs to have $p+q-1$ edges to connect $p+q$ cycles. Moreover, every cycle of size m such that $m \geq 3$ will have m edges on it. So the total number of edges in the graph is $|E| = p + q - 1 + \sum_{i=1}^{q} m_i$, where m_i is the number of vertices on the cycle C_i. The quantity $p + \sum_{i=1}^{q} m_i$ is equal to the total number of vertices in the graph. Hence $|E| = q - 1 + |V|$. On the other hand, the smallest size of a cycle is 3, therefore we conclude that $q \leq \lfloor \frac{n}{3} \rfloor$ which implies that $|E| \leq \lfloor \frac{4n}{3} \rfloor$.

Theorem 11. *The complexity of the $BroadcastTC(G, v)$ on a graph $TC = (V, E)$ is $O(n)$ where $|V| = n$.*

Proof. The proof will be done by induction. First consider a leaf cycle which is the base case of the algorithm. The broadcast problem in a simple cycle has a linear complexity.

At every recursive call of the algorithm one cycle will be considered. Consider a cycle C_l, assume that the broadcast time and scheme of each subgraph, T_{TC_j} $1 \leq j \leq p$, obtained by removing the edges E_{C_l}, $|E_{C_l}| = p$, can be calculated in linear time. The calculation of the set E_{C_l} has complexity $O(p)$ and can be done by traversing the cycle and inspecting all the edges having an end point on the cycle.

The next step is to construct the unicyclic graph, G_{C_l}, which consists of the cycle C_l attached to it the broadcast trees T_{TC_j}, $1 \leq j \leq p$, where T_{TC_j} is the broadcast tree of the subgraph TC_j. We need to prove that the broadcast time and scheme of this graph can be calculated in linear time. Given the results proved in the previous chapter ([67]) we can calculate the broadcast time in $O(p)$ time because the broadcast times of the subgraphs are known. The unicyclic broadcast algorithm uses the already calculated broadcast time of each of the graphs TC_i, $1 \leq i \leq p$ and it has to construct the l spanning trees of the graph G_{C_l} to find the one which has the least broadcast time. Calculation of the first spanning tree has complexity $O(l)$ where l is the length of the cycle. The calculation of each of the remaining $l-1$ trees has a constant time complexity. Therefore, the total complexity of the algorithm is $O(|TC_1|) + \cdots + O(|TC_p|) + O(p) + O(l)$ which is equal to $O(|E|)$, where E is the set of edges of the graph G_{C_l} and $|TC_j|$ is the number of vertices of the tree of cycles TC_j. Since $|E| \leq \lfloor \frac{4n}{3} \rfloor$ we conclude that the complexity of the algorithm is $O(|V|)$. □

Consider a graph G that has a bridge (v_1, v_2) (Fig. 6). In other words, the removal of the edge (v_1, v_2) divides the graph G into two connected components G_1 and G_2. Without loss of generality, assume that G_1 is the component containing the originator v_o. We will prove that there exists a divide and conquer approach for solving the broadcast problem in the graph G. First solve the broadcast problem in G_2 assuming that the originator is v_2. This results in a broadcast tree, a spanning tree of G_2, which can be substituted instead of G_2 without changing the broadcast time of v_o in the graph G. We will denote by G' the graph G_1 connected to T_2, where T_2 is the broadcast tree of G_2 (Fig. 7).

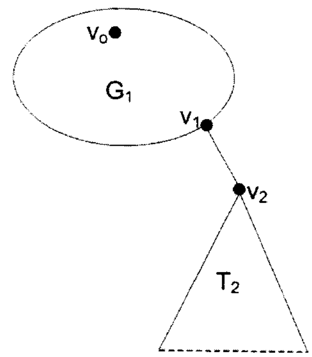

Figure 7: The graph where G_2 is substituted by its broadcast tree T_2.

Theorem 12. *Consider a graph G with a bridge (v_1, v_2) which divides G into two components G_1 and G_2. Let the graph G' be constructed from G such that the broadcast tree of G_2 substitutes the graph G_2 (as in Figures 6 and 7). For any originator v_o in G_1 the broadcast time $b(v_o, G) = b(v_o, G')$.*

Proof. Let S be the optimal broadcast scheme in G, and S' be the optimal broadcast scheme in G'. In both schemes when v_2 gets informed it does not have any uninformed neighbours in G_1 and all its uninformed neighbours are in G_2. Therefore, the broadcast schedule that v_2 chooses does not affect on the time at which the rest of the vertices in G_1 get informed. Two cases might arrise:

1. According to scheme S there is a vertex v in G_1 that gets informed at time $b(v_o, G)$. Assume that t_2 is the time at which v_2 was informed, therefore all the vertices of G_2 were informed within $b(v_o, G) - t_2$ time units after v_2 was informed. In scheme S' vertex v will still be informed at time $b(v_o, G)$ because it is in G_1 and changing the graph structure of G_2 will not affect on the broadcast schedule of the vertices in G_1. Moreover, all the vertices of G_2 will be informed at time $t_2 + b(v_2, G_2)$. Since

$b(v_2, G_2) \leq b(v_o, G) - t_2$ then the broadcast time $b(v_o, G) = b(v_o, G')$.

2. According to scheme S there is no vertex in G_1 that gets informed at time $b(v_o, G)$. Therefore, there is a vertex v'_f in G_2 that gets informed at time $b(v_o, G)$. Since G' is a subgraph of G then we know that $b(v_o, G') \geq b(v_o, G)$. Assume that $b(v_o, G') > b(v_o, G)$. Consider a new scheme S'' in G' such that every vertex in G_1 keeps the same broadcast schedule as in S, while every vertex in G_2 keeps the same broadcast schedule of S'. In scheme S let t_2 be the time at which v_2 gets informed and let t_{G_2} be the time needed for the last vertex in G_2 to be informed after v_2 gets informed, hence $b(v_o, G) = t_2 + t_{G_2}$. In S'' vertex v_2 will again be informed at the same time t_2, therefore the time at which the last vertex in G_2 will be informed is $b(v_o, G') = t_2 + b(v_2, G_2)$. Since $b(v_2, G_2) \leq t_{G_2}$ then we conclude that $b(v_o, G') \leq b(v_o, G)$ which contradicts our assumption that $b(v_o, G') > b(v_o, G)$. Therefore $b(v_o, G') = b(v_o, G)$.

□

Theorem 12 presents the result that is necessary to prove the correctness of the broadcast algorithm described in the previous section. This result together with the correctness of the broadcast algorithm of unicyclic graphs imply the correctness of the above algorithm.

Theorem 13. *The $BroadcastTC$ algorithm calculates the optimal broadcast scheme and the broadcast time of any originator in any tree of cycles TC.*

Proof. We will prove this theorem by induction. Consider the base case where a cycle C_q is considered such that all of the subgraphs obtained by cutting the edges E_{C_q} are simple cycles. The broadcast time and scheme of the simple cycles are correctly calculated by cutting the $\lceil \frac{n}{2} \rceil$-th edge from the originator. According to Theorem 12, we can substitute the broadcast trees of each of the cycles without changing the broadcast time and scheme of any originator on the cycle C_q. Assume that the algorithm is correct for any subgraph of the tree of cycles. Now consider the originator, v_o, belonging to the cycle C_o. We are assuming that the broadcast times and schemes of all the subgraphs obtained by removing the edges in E_{C_o} have been correctly calculated. The inductive step is to prove that the algorithm correctly calculates the broadcast time of the TC containing the cycle C_o, together with the edges E_{C_o}, and the connected subgraphs. Because of Theorem 12, we can use the broadcast trees of each of the subgraphs to obtain a unicyclic graph with the cycle being C_o whose broadcast time and scheme is the same as that of TC. The correctness of the broadcast algorithm in unicyclic graphs implies that the above algorithm correctly calculates the broadcast time of any tree of cycles. □

4.2 Necklace Graphs

A necklace graph is a graph with m cycles and $m - 1$ vertices such that each of these vertices is common to two cycles (see Figure 11).

Definition 5. *Given m cycles $C_i = (V_i, E_i)$ such that $V_i \cap V_{i+1} = \{v_i\}$ for $1 \leq i \leq m-1$, a necklace graph is defined to be the graph $G = (V, E)$ where $V = V_1 \cup \cdots \cup V_m$ and $E = E_1 \cup \cdots \cup E_m$.*

We will present an algorithm to solve the broadcast problem in an arbitrary necklace graph [63]. First, we will consider the case where the originator belongs to one of the end cycles. The solution for the case when the originator is not on an end cycle can be obtained if one can solve the first case (i.e. the originator is on an end cycle). Assume that the originator is in cycle C_i and the end cycles are cycles C_1 and C_k. Two necklace graphs can be constructed one being the subgraph starting at cycle C_1 and ending at cycle C_{i-1}, and the other one being the subgraph starting at C_{i+1} and ending at C_k. After solving the broadcast problem in these two graphs the solutions can be put together to construct the solution for the original graph.

4.2.1 Cycle and an Attached Graph

In this section we will present some results which will be helpful in proving the correctness of the algorithm we propose. First we consider a graph G made up of a cycle C_N and a graph G' such that both graphs have a vertex v_c in common. We call the vertex v_c the connecting vertex.

Theorem 14. *For any originator v_o on C_N, there exists an optimal broadcast scheme which first sends the information along the shorter path towards vertex v_c and then along the longer path.*

Proof. Assume that there is no optimal broadcast scheme such that v_o first sends the information along the shortest path towards the connecting vertex v_c. Let S be one of the optimal broadcast schemes. We will construct a new scheme S' which will first send the information towards the shortest path and we will show that it has a broadcast time smaller than or equal to that of S.

Consider the scheme S, according to this scheme the connecting vertex v_c gets informed at a certain time which we label by t_1. After it gets informed, it spends b time units informing vertices in G' before it forwards the message on C_N. At time $t_1 + b$ there will be $2t_1 + b$ informed vertices (including the originator) on the cycle and a chain of $N - 2t_1 - b$ uninformed vertices on C_N with both ends of the chain connected to informed vertices. We know that these vertices will be informed at time $b(G)$ hence we can conclude that $(N - 2t_1 - b)/2 \leq b(G) - t_1 - b$. We will construct a new scheme S' from S which is very similar to S with the only difference that v_0 first informs the vertex on the cycle that is closer to v_c. Similar to scheme S, when v_c gets informed, it will first spend b time units informing the same vertices in G' (and in the same order). Now we need to calculate the broadcast time of scheme S' and compare it with that of S. According to scheme S' vertex v_c will get informed at time $t_1 - 1$. Moreover since v_c keeps the same broadcast schedule for informing the vertices of G', the vertices of G' will be informed in at most $b(G) - t_1$ time units. Therefore, in scheme S' at time $(t_1 - 1) + (b(G) - t_1) = b(G) - 1$ the vertices of G' will be informed. On the other hand, the number of informed vertices on the cycle at time $t_1 + b$ can be calculated by adding the following:

1. All the $t_1 - 1$ vertices on the shorter path between v_o and v_c.

2. The originator v_0.

3. $t_1 - 1 + b$ informed vertices on the longer path between v_o and v_c.

4. One vertex neighbouring v_c on the longer path between v_c and v_o.

The sum of the informed vertices is $2t_1 + b$ which implies the number of uninformed vertices is $N - (2t_1 + b)$ which is the same as in scheme S. Therefore, all the vertices will be informed at most by time $b(G)$. Also we can conclude that if $(N - 2t_1 + b)/2 < b(G) - t_1 - b - 1$, then broadcast time of scheme S will be $b(G) - 1$. This contradicts the assumption according to which there was no optimal scheme which first informs the vertex on the shorter path to the connecting vertex. □

The broadcast time of a cycle with a tree

In this subsection we will calculate the broadcast time of a graph composed of a cycle with a tree T is attached to one of its vertices, such that the root of the tree T is a vertex on the cycle. We are limiting the analysis to the case where the root of the tree has degree 2. Consider a graph G made up of a cycle C_N and a tree T connected to vertex v_m 8. Let v_o be the originator. According to theorem 14 an optimum broadcast scheme can be obtained by first informing the vertex on the shortest path from v_o to v_m. After that, the broadcast scheme is simple up to the time when vertex v_m is informed since every informed vertex informs its only uninformed neighbor which is also a vertex on the cycle. The only vertex that has more than one choice is v_m. Let v'_m be the other vertex that was informed at the same time as v_m. Let P be the number uninformed vertices on the cycle C_N at the time v_m was informed.

Assume that the broadcast time of v_m in the tree T is equal to t, i.e. $b(v_m, T) = t$. Let T_1 and T_2 rooted at v_1 and v_2 respectively, be the two subtrees connected to v_m, such that the broadcast time of v_1 is T_1 is equal to t_1 and the broadcast time of v_2 in T_2 is t_2 (Fig. 8). The broadcast time of G depends on the values of t, t_1, t_2 and P.

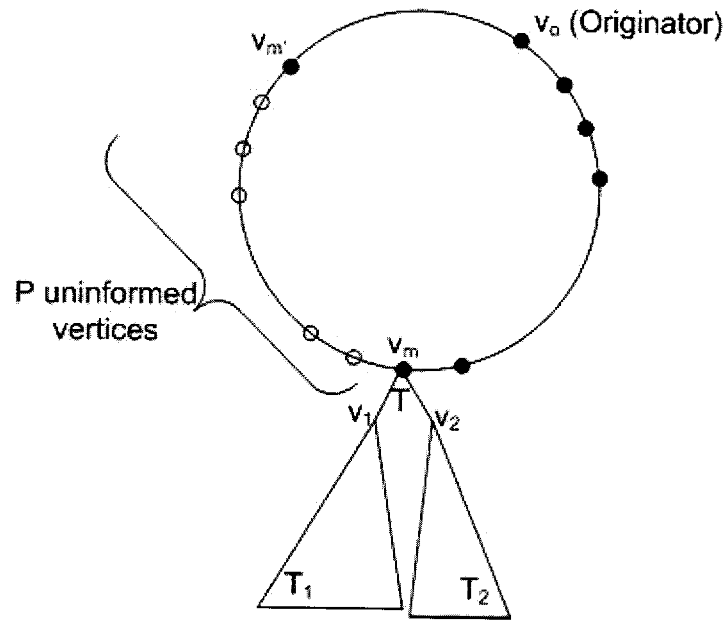

Figure 8: Shows a tree attached to a cycle.v_o is the originator and the P vertices on the cycle between v_m and $v_{m'}$ are uninformed.

1. $t < \lceil \frac{P}{2} \rceil$: The broadcast time of G will be exactly the same as that of C_N. Therefore, An optimum broadcast scheme is obtained when the vertex v_m first informs the vertex on the cycle C_N and then starts informing the vertices in the attached tree using the tree broadcast algorithm of [112]

2. $t > \lceil \frac{P}{2} \rceil$: The broadcast time of G will be equal to the time to inform v_m plus the time needed to broadcast the information in the tree T. $b(G) = d(v_o, v_m) + t$, where $d(v_o, v_m)$ is the distance between vertices v_o and v_m. An optimum broadcast scheme can be obtained if v_m spends the next two time units informing the vertices in the attached tree and then forwards the message on the cycle C_N.

3. $t = \lceil \frac{P}{2} \rceil$: This case needs a careful observation because the structure of the tree also

has a role in determining the broadcast time of the graph G. The broadcast scheme will be different depending on which one of the following cases applies:

i– P is even, $b(T) = t = P/2, b(T_1) = t-1, b(T_2) \leq t-3$.

ii– P is even, $b(T) = t = P/2, b(T_1) = t-1, b(T_2) = t-2$.

iii– P is even, $b(T) = t = P/2, b(T_1) = t-2, b(T_2) = t-2$.

iv– P is odd, $b(T) = t, b(T_1) = t-1, b(T_2) = t-2$.

v– P is odd, $b(T) = t, b(T_1) = t-2, b(T_2) = t-2$.

For all these cases, inform the vertex on the cycle first, then T_1, and finally T_2, the broadcast time of this scheme will be $b(G) = d(v_o, v_m) + t + 1$. Actually informing T_1 first then the vertex on the cycle and finally T_2 will yield the same broadcast time. The rest of the permutations will all yield a broadcast time greater than or equal to $d(v_o, v_m) + t + 1$. Now let us study the broadcast tree obtained by the first scheme. Since v_m will inform first a vertex on the cycle, then the broadcast tree is the tree obtained by cutting the edge between the vertices at distance $P/2$ and $P/2+1$ from v_m. Note that the root of the resulting broadcast tree has degree 2 hence there are two subtrees connected to the root. The subtree containing v'_m is a simple path of length $d(v_o, v_m) - 2 + (P/2 - 1)$ which has a broadcast time of $d(v_o, v_m) + P/2 - 3$. The subtree containing v_m has a broadcast time of $d(v_o, v_m) + P/2 - 1$ which is 2 more than the broadcast time of the other subtree. It is worth comparing this broadcast tree with the tree obtained by the second broadcast scheme. The main difference is that the second broadcast tree will have two subtrees such that the difference between the broadcast times of the two trees is 1.

vi– P is odd, $b(T) = t = (P+1)/2$, $b(T_1) = t-1$, $b(T_2) \leq t-3$. First inform the root of T_1, then inform the vertex on the cycle, and finally inform the root of T_2. The broadcast time of this scheme will be $d(v_o, v_m) + t$. Spending the first time unit informing the root of tree T_1 will guarantee that subtree T_1 will be informed in t time units. After informing the root of T_1 in the first time unit, v_m will inform its neighbor on the cycle. Therefore, in the next $t-1$ time units $t-1$ vertices on the cycle will receive the message that was sent by v_m. The remaining t vertices on the cycle will be informed by v'_m. Since $b(T_2) \leq t-3$, if v_m informs the root of T_2 at the third time unit the broadcast time of T will still be t. The broadcast tree of the graph will have two subtrees rooted at v_o both having a broadcast time of $d(v_o, v_m) + t - 2$. Note that any other scheme will have a higher broadcast time. Using the six cases presented above we can conclude that only in one case (the 6th case) it is possible to have a broadcast time of $d(v_o, v_m) + \lceil \frac{P}{2} \rceil$. For the other cases the broadcast time is 1 more than this value.

To sum up, we can say that given a graph made up of a tree attached to a cycle, the broadcast time of the graph depends on two factors: the broadcast time of each individual component(the cycle and the tree) and the structure of the tree. The broadcast time of the tree is certainly important for obvious reasons but in the discussion above we saw that the structure of the tree plays a role too in determining the broadcast time of the whole graph. For example it is possible to have a cycle C_p and two trees T_1 and T_2 with equal broadcast times such that the broadcast time of the graph C_p attached to T_1 is different than that of the graph C_p attached to T_2.

In general, dividing a graph into smaller subgraphs and solving the broadcast problem in the smaller graphs and joining the solutions does not always yield an optimum result. However, this is possible in necklace graphs provided that some care is taken while choosing the broadcast tree in case there is more than one possibility. The following observations can be made based on the previous results.

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

Lemma 5. *For any originator v_o on C other than v_c the broadcast time $b(v_o, G) = b(v_o, G_T)$ where the graph G_T is made up of the cycle C and the broadcast tree T. If there are several broadcast trees then the tree that has $b(v_1, T_1) \geq b(v_2, T_2) + 2$ must be chosen.*

Finally note that all the broadcast trees of a necklace graph have roots of degree 2 because they are on a cycle where all vertices have degree 2.

4.2.2 A Broadcast Algorithm in Necklace Graphs

Using the previous results, we can construct a spanning tree which will yield the optimal broadcast time for the necklace chain graph, N_k, and be its broadcast graph. The broadcast tree will be constructed using a bottom-up approach. In the algorithm presented below we will denote subgraphs of the necklace graphs, which are themselves necklace graphs, by $N_{i,j}$ if the subgraph has cycle C_i as its first cycle and cycle C_j as its last cycle. We will also make use of the algorithm presented above to calculate the broadcast time of a graph which consists of a cycle and a tree attached to it. We will name this algorithm $CycleTreeAlgorithm(G, v_o)$.

Broadcast algorithm:
Input: A necklace graph N_k and an originator v_o in C_1
Output: The Broadcast time and scheme of N_k
T = the broadcast tree of the cycle C_k ;
for $i \leftarrow k-1$ ***to 1*** **do**
  G = the cycle C_i with the attached tree T ;
  if $i = 1$ **then**
    $v_o' = v_o$
  else
    v_o' = vertex joining C_i to C_{i-1}
  end
  T = CycleTreeAlgorithm(G,v_o')
end
return T

The tree construction is basically is the process of removing an edge in every cycle. The construction of the broadcast tree in the bottom-up approach starts by cutting the furthest cycle at the mid point. If there are odd number of vertices on the cycle then the resulting tree will have two branches of the same size, otherwise one branch will have one more vertex. The broadcast time of the last cycle is $\lceil \frac{P}{2} \rceil$, where P is the number of vertices on the last cycle. In the next step move one cycle closer to the originator. We will have one cycle and the tree we just constructed. Solving the broadcast problem of such a graph was the subject of the previous sections. Moreover, we argued that the broadcast tree of this graph will also be a broadcast scheme of the 2 cycles. The construction proceeds this way until we get a broadcast tree where the root is the originator.

Finally we show that the complexity of the algorithm is linear. First of all, finding the last cycle takes a linear time. Next we build the broadcast tree in a bottom up approach and calculate the broadcast time of the root. For the construction of a broadcast tree of a graph made up of a cycle with an attached tree to one of its vertices all the information about the tree has already been calculated. The only remaining task can be solved by traversing the cycle once. This will take a linear time. Hence the complexity of the whole algorithm is: $O(N_1) + O(N_2) + \cdots + O(N_k)$ where N_i, $1 \leq i \leq k$ is the size of the cycles C_i.

4.2.3 Broadcasting from an Internal Cycle

In this section we will study the case where the originator in the necklace graph is not on an end cycle but is on an internal cycle. The approach we use to solve this is by dividing the problem into two instances where in each instance the originator is on an end cycle. After the problem is solved in these two instances the solutions will be joined to obtain the broadcast time and broadcast scheme for the original problem.

Assume that we are given a necklace graph with k cycles and an originator v_o on cycle C_i.We will split the necklace chain into into 2 and solve the broadcast problem in each one of them. In each problem the originator is on an end cycle. The first necklace graph is made up of the cycles C_1 to C_{i-1} and the second necklace graph is made up of the cycles C_{i+1} to C_k. In the first graph the originator will be considered to be the connecting vertex which connects the cycles C_{i-1} and C_i. For the second graph the originator is considered to be the vertex connecting the cycles C_i and C_{i+1}. After solving the broadcast problem in the two necklace graphs we get 2 broadcast trees which when attached back to the cycle C_i result in a unicyclic graph. The final step is to solve the broadcast problem in the unicyclic graph where the originator is v_o.

Earlier we argued that it is possible to split a necklace graph into two, solve the problem in one part and use its broadcast tree to solve the broadcast problem for the whole Necklace graph. The case that we face now is quite similar but with two necklace graphs connected at two different vertices of a cycle. Let us denote the cycle by C and the first necklace graph by N_1 which is connected to C at vertex v_1. The second necklace graph, N_2, is connected to the cycle at vertex v_2. Let G denote the graph make up of the two Necklace graphs and the cycle (i.e. the complete necklace graph). We will prove that there exists a broadcast tree in G which has two subtrees which are the broadcast trees of N_1 and N_2.

Theorem 15. *There exists a broadcast tree in G such that the subtree induced by the vertices of N_1 and N_2 are broadcast trees for N_1 and N_2 respectively.*

Proof. Assume that there does not exist a broadcast tree of G where the trees T_1 and T_2 are both broadcast trees of N_1 and N_2 respectively. Therefore, there exits a broadcast tree of G such that at least one of the trees, T_1 and T_2, induced by N_1 and N_2 is not a broadcast tree of N_1 or N_2. Let T be a broadcast tree of G. Without loss of generality assume that T_1 is not a broadcast tree of N_1. The vertex which connects the necklace graph N_1 to the cycle C is labeled by v_1. The optimal broadcast scheme that generates T can have one of the 3 possible broadcast schedules for vertex v_1.

1. v_1 informs a vertex on the cycle C_1 first and then the next 2 time units informs its neighbors in N_1. In this case, we can modify the optimal broadcast tree T of G by substituting the tree T_1 with an optimal broadcast tree of N_1. The resulting broadcast tree will still be an optimal broadcast tree.

2. v_1 informs its two neighbors in N_1 in the first 2 time units then informs its neighbor on C. Again we can modify the optimal broadcast tree T of G by substituting T_1 with an optimal broadcast tree of N_1. The resulting tree will be an optimal broadcast tree.

3. v_1 first informs a neighbor in N_1 in the first time unit after it gets informed. In the second time unit, it informs its neighbor on C and in the third time unit it informs its second neighbor in N_1. Since we know that T_1 is not a broadcast tree of N_1, then $b(v_1, T_1) > b(v, N_1)$. Another conclusion we can draw from the given assumptions is that after v_1 gets informed it takes at least $b(v_1, T_1)$ time units for all the vertices of N_1 to be informed. Therefore, we can build another broadcast tree/scheme by using the optimal broadcast tree of N_1 and changing the broadcast scheme at v_1 such that v_1 informs its neighbor on C first after it gets informed and in the second and third time units it informs the vertices of the broadcast tree of N_1. This broadcast tree will inform all the vertices of N_1 in $b(v_1, N_1) + 1$ time units after v_1 gets informed. Note that $b(v_1, N_1) + 1 \leq b(v_1, T_1)$ therefore the new broadcast scheme is also an optimal broadcast scheme and the tree is a broadcast tree.

If T_2 is not a broadcast tree of N_2 we can do the same substitution and obtain a new broadcast tree for the graph G which contains subtrees which are themselves broadcast trees of N_1 and N_2. Hence we arrive at a contradiction and our assumption that there does not exist a broadcast tree with subtrees that are broadcast trees of N_1 and N_2 was false. □

As a result of the theorem proved above we conclude that it is possible to build a broadcast tree of the graph G by first solving the broadcast problem in the two necklace graphs and then use the broadcast trees of those two graphs. However, it is possible that a necklace graphs N_1 and N_2 can have more than one broadcast tree. The theorem proved above does not guide us in choosing the correct broadcast tree for each necklace graph. However, we already discussed above the case where we had only one tree attached to a cycle. Note that we are considering broadcast trees where the root has degree 2. In other words the root has 2 subtrees attached to it. For the case of 1 tree attached to a cycle, our conclusion was to choose a tree such that the difference in the broadcast time of the subtrees attached to the root was as big as possible (actually greater than or equal to 2 is enough). The same conclusion applies to the current situation too. The justification for this lies in the fact that if in a broadcast tree one of the subtrees has a broadcast time which is 2 (or more) time units less than the other subtree, then the root vertex connecting the necklace graph to the cycle can first inform a vertex in the necklace graph and then inform its neighbor on the cycle in the second time unit and inform its neighbor in the necklace graph at the third time unit without increasing the broadcast time of the attached necklace graph.

4.2.4 Broadcasting from Connecting Vertices

A connecting vertex is a vertex that is common to two cycles. In the above algorithm we assumed that the originator is not a connecting vertex. The correctness of the algorithm depends on Theorem 15 which assumes that when the connecting vertex is informed one of its neighbors is informed too. In this part we study the case where the originator is a connecting vertex. The connecting vertex v_c is connecting two cycles and hence has a degree 2 in each cycle.

First we will prove a general result that is not limited to cycles and then we will apply that to solve the broadcast problem in necklace graphs when the originator is a connecting vertex. Consider a graph $G = (V, E)$ which can be divided into two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that $V_1 \cap V_2 = \{v_c\}$, $V_1 \cup V_2 = V$, $E_1 \cap E_2 = \emptyset$, and $E_1 \cup E_2 = E$. In other words v_c is a cut vertex. Moreover, the graphs G_1 and G_2 are such that the degree of v_c is 2 in both graphs. Let T_1 rooted at v_c be a broadcast tree of G_1 and T_2 be a broadcast tree of G_2. Since v_c has degree 2 in each graph then T_1 has two subtrees connected to v_c which we denote by T_a and T_b rooted at v_a and v_b respectively. Similarly, T_2 has two subtrees T_c and T_d rooted at v_c and v_d.

Theorem 16. *Given a graph G with cut vertex v_c as defined above, if $b(v_c, G_1) \neq b(v_c, G_2)$ then a broadcast tree of the originator v_c in G can be obtained by constructing the tree T rooted at v_c by connecting v_c to the vertices v_a, v_b, v_c, and v_d the roots of the subtrees T_a, T_b, T_c, and T_d respectively.*

Proof. Without loss of generality assume that $b(v_c, G_1) > b(v_c, G_2)$. Also we assume that $b(v_a, T_a) \geq b(v_b, T_b)$ and $b(v_c, T_c) \geq b(v_d, T_d)$. In order to prove the theorem by contradiction, assume that there exists another tree T' such that $b(v_c, T') < b(v_c, T)$ and one of these conditions apply:

1. The subtree T_1' induced by G_1 is not a broadcast tree of G_1.

2. The subtree T_2' induced by G_2 is not a broadcast tree of G_2.

3. Both subtrees T_1' and T_2' are not broadcast trees of G_1 and G_2.

If T_1' is not a broadcast tree of G_1 then $b(v_c, T_1') > b(v_c, T_1)$ which in turn implies that $b(v_c, G) \geq b(v_c, G_1) + 1$. If $b(v_c, G) \geq b(v_c, G_1) + 2$ then we can easily substitute T_1' by T_1 and still have $b(v_c, T) = b(v_c, T') \geq b(v_c, G_1) + 2$. In this case the broadcast scheme of v_c will be to inform its children in the following order: v_a,v_b, v_c, and v_d. Therefore it must be that $b(v_c, G) = b(v_c, G_1) + 1$. In this case we know that $b(v_c, T_1) < b(v_c, T_1')$ which implies that if we substitute T_1' by T_1 and keep the broadcast schedule of v_c unmodified, the broadcast time $b(v_c, T) = b(v_c, T')$ which is a contradiction.

The rest of the cases can be analyzed similarly and all of them will lead to a contradiction meaning that there cannot be another tree T' such that $b(v_c, T') < b(v_c, T)$ where the tree T is constructed from the broadcast trees of G_1 and G_2 as mentioned above. $\square$

The case where $b(v_c, G_1) = b(v_c, G_2)$ shows a different behaviour. Note that in this case $b(v_c, G) > b(v_c, G_1)$ and there are two scenarios to study:

1. $b(v_c, G) = b(v_c, G_1) + 1$

2. $b(v_c, G) \geq b(v_c, G_1) + 2$.

In the second case it is easy to see that using the optimal solutions of G_1 and G_2 always guarantees the optimal solutions of G. However, in the first case a non-optimal broadcast tree of one of the graphs combined with an optimal broadcast tree of the other graph can result in the optimal solution of G. In order to show this, we need to exhaustively list all the possible values of the broadcast times of two spanning trees of G_1 and G_2. Let T_1 represent a spanning tree of G_1, the two subtrees of T_1 are denoted by T_a and T_b rooted at v_a and v_b. Similarly, T_2 is a spanning tree of G_2 and its two subtrees are T_c and T_d rooted at v_c and v_d.

We will label the value of $b(v_c, G_1)$ by τ so given the assumption that $b(v_c, G) = \tau + 1$ we can have the following values:

1. $b(v_a, T_a) = \tau - 1, b(v_b, T_b) \leq \tau - 3, b(v_c, T_c) = \tau - 1, b(v_d, T_d) \leq \tau - 3.$

2. $b(v_a, T_a) = \tau, b(v_b, T_b) \leq \tau - 3, b(v_c, T_c) = \tau - 1, b(v_d, T_d) \leq \tau - 2.$

3. $b(v_a, T_a) = \tau, b(v_b, T_b) \leq \tau - 2, b(v_c, T_c) = \tau - 1, b(v_d, T_d) \leq \tau - 3.$

4. $b(v_a, T_a) = \tau - 1, b(v_b, T_b) \leq \tau - 3, b(v_c, T_c) = \tau, b(v_d, T_d) \leq \tau - 2.$

5. $b(v_a, T_a) = \tau - 1, b(v_b, T_b) \leq \tau - 2, b(v_c, T_c) = \tau, b(v_d, T_d) \leq \tau - 3.$

Note that the first element of the list implies that $b(v_c, T_1) = b(v_c, G_1)$ and $b(v_c, T_2) = b(v_c, G_2)$. However, the second and third entry in the above list imply that $b(v_c, T_1) = \tau + 1$ which implies that T_1 is not a broadcast tree of G_1 even though the tree constructed by T_1 and T_2 is an optimal tree. A similar conclusion that T_2 is not a broadcast tree of G_2 can be obtained from the fourth and fifth elements in the list.

So far we can conclude that it is possible to have an optimal solution for G using non optimal broadcast trees from either G_1 or G_2. Next we will show that there are cases where optimal solutions from G_1 and G_2 can result in a non-optimal solution for G. Consider the following list:

1. $b(v_a, T_a) = \tau - 1$, $b(v_b, T_b) \leq \tau - 2$, $b(v_c, T_c) = \tau - 1$, $b(v_d, T_d) \leq \tau - 2$.

2. $b(v_a, T_a) = \tau - 2$, $b(v_b, T_b) \leq \tau - 2$, $b(v_c, T_c) = \tau - 1$, $b(v_d, T_d) \leq \tau - 2$.

3. $b(v_a, T_a) = \tau - 1$, $b(v_b, T_b) \leq \tau - 2$, $b(v_c, T_c) = \tau - 2$, $b(v_d, T_d) \leq \tau - 2$.

In all the above combinations, both T_1 and T_2 are optimal broadcast trees but the broadcast time of T will be $\tau + 2$. Therefore, we can conclude that if $b(v_c, G_1) = b(v_c, G_2)$ then it is possible to have $b(v_c, G) = \tau + 1$ by using non-optimal broadcast trees from G_1 or G_2.

We can use this result to study the broadcast problem in necklace graphs where the originator is a connecting vertex. A simple algorithm will be to first study the problem in the two parts of the necklace graph N_1 and N_2. If $b(v_o, N_1) \neq b(v_o, N_2)$ then the broadcast trees of N_1 and N_2 are used to construct a tree T and calculate the broadcast time of T using the well known broadcast algorithm for trees. However, if $b(v_o, N_1) = b(v_o, N_2)$ then the bottom up algorithm should be modified such that at every iteration two spanning trees should be calculated. The first tree is the broadcast tree which has the maximum imbalance between its two subtrees and the second one is a spanning tree whose broadcast time is one more than the broadcast time of the optimal tree and again has the maximum imbalance between its two subtrees.

It remains to show how to construct spanning trees in necklace graphs where the broadcast time of the originator is one more than the optimal broadcast time. Consider the example we presented above which is a cycle connected to a tree. If we show how to construct such a spanning tree in this graph we, the same idea can be applied in every iteration of the algorithm to obtain such a spanning tree for the whole necklace graph. Using the same notations as above, (t is the broadcast time of the tree, and P is the number of uninformed vertices on the cycle when the connecting vertex is informed) we can analyze the same cases as before:

1. If $t < \lceil \frac{P}{2} \rceil$, then the cycle is the one that determines the broadcast time. It is always possible to get a spanning tree in a cycle such that the broadcast time is one more than the optimal broadcast time. The difference in the broadcast time of the two branches of the spanning tree can be at least 3 if the cycle has even number of vertices and maximum of 4 if the cycle had odd number of vertices. Note that it is impossible to use have a different tree T' with broadcast time equal to $\lceil \frac{P}{2} \rceil$ and get a spanning tree for the graph with broadcast time one more than the optimal broadcast time, and the difference in the broadcast time of the two branches greater than 3.

2. If $t > \lceil \frac{P}{2} \rceil$, then the tree determines the broadcast time of the graph. Having a tree with broadcast time $t + 1$ guarantees that we can get a spanning tree with broadcast time one more than the optimal broadcast time. Any tree with broadcast time greater than $t + 1$ will result in a bigger broadcast time which we are not interested in.

3. If $t = \lceil \frac{P}{2} \rceil$, in this case we need to consider all the six cases as it was done before. For all the 3 cases when P is even it is possible to obtain a spanning tree in the graph with broadcast time one more than the optimal broadcast time by the broadcast scheme where the message which was forwarded by the connecting vertex is sent up

on the cycle for one more time unit. Hence the difference in the broadcast time of the two branches of the spanning tree will be 3. If a different tree T' was attached to the cycle with broadcast time $t + 1$ it would still be possible to have a broadcast time of $t + 1$ in the spanning tree but the difference in the broadcast times of the two branches will be less than 3. Finally, if P is odd and $t_1 = t - 1$ and $t_2 \leq t - 3$ then if the connecting vertex first informs its neighbor on the cycle then the broadcast time of that scheme will be 1 more than the optimal scheme and the difference of the broadcast time of the two branches of the spanning tree will be 4. Any other tree attached to the cycle with time $t + 1$ will give a broadcast time of the graph $t + 1$ but the difference in the broadcast time of the branches of the spanning tree will be less than 4. The other combinations of t_1 and t_2 when P is odd are similar to the case where P is even.

4.2.5 Upper and Lower Bounds

Consider a broadcast algorithm which tries to inform the vertices of the next cycle whenever possible. Such a scheme is a greedy scheme. We will show that the greedy scheme cannot always generate the optimal solution. However, we will use this scheme to prove that the upper bound on the broadcast time of a Necklace graph is $d + 2$. We will also give an example of a necklace graph whose broadcast time is indeed equal to $d + 2$ hence proving that $d + 2$ is a tight upper bound.

The lower bound on the broadcast time can be obtained easily. We know that for any graph G, $b(G) \geq d$ where d is the diameter of the graph G. Figure 9 shows a necklace graph

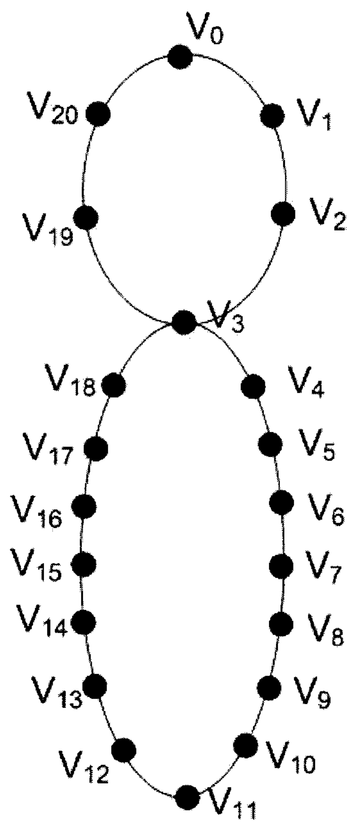

Figure 9: A necklace graph with 2 cycles whose diameter is 11. The broadcast time of this graph is 11 too.

whose diameter is equal to 11 which is the length of the distance between vertices v_0 and v_{11}. One can easily see that the broadcast time of this necklace graph is equal to 11 which is the broadcast time of vertices v_0 and v_{11}. Therefore we can conclude that $b(G) \geq d$ is a tight lower bound for necklace graphs.

In the remaining part of this section we will study the upper bound of necklace graph. First we describe a greedy broadcast algorithm on necklace graphs. Assume that we are given a necklace graph with k cycles and an originator v_o which is on the end cycle C_0. The greedy algorithm, $A_{GreedyNecklace}$, is as follows:

1. v_o first sends the message to the neighboring vertex that is closest to C_1. In the second time unit v_o informs its other neighbor.

2. If an informed vertex is not a vertex that connects two different cycles, then it informs its neighbour at the next time unit after it gets informed.

3. If an informed vertex,v, belongs to 2 cycles C_j and C_{j+1}, it has 3 uninformed neighbors to inform. Assuming that the vertex informing v was in C_j, v first informs its neighbour on C_{j+1} that is closest to C_{j+2}, then it informs the other vertex on C_{j+1} and finally informs its uninformed neibhour on C_j.

Let v_o be the originator on an end cycle and v_{c_i} be the vertex that connects cycles C_{i-1} and C_i.

Theorem 17. *In the greedy algorithm $A_{GreedyNecklace}$ the time it takes to inform vertex v_{c_i} is equal to the distance $d(v_o, v_{c_i})$.*

Proof. We can prove this theorem by induction. Consider the first connection vertex connecting the cycles C_0 and C_1. According to the algorithm the originator first informs its neighbour that is closest to v_{c_1}. Moreover, every newly informed vertex on the path between v_o and v_{c_1} informs its uninformed neighbour in the next time unit. Therefore, v_{c_1} will be informed in $d(v_0, v_{c_1})$ time units. Hence we proved the base case of the induction.

Assume that the theorem is true for c_{v_i} we need to prove that the result will be correct for $v_{c_{i+1}}$. Because of the assumption we know that v_{c_i} was informed in $d(v_o, v_{c_i})$ time units. Note that v_{c_i} belongs to the two cycles C_{i-1} and C_i. According to the description of the greedy algorithm when vertex v_{c_i} is informed, it first informs its two neighbours that are on the cycle C_i and then informs its uninformed neighbour on C_{j-1}. Therefore, vertex $v_{c_{i+1}}$ will be informed $d(v_{c_i}, v_{c_{i+1}})$ time units after v_{c_i} gets informed. Hence, we conclude that $v_{c_{i+1}}$ gets informed at time $d(v_o, v_{c_{i+1}})$. This proves the inductive step and concludes the proof. □

Theorem 18. *The greedy algorithm $A_{GreedyNecklace}$ finishes broadcasting in $d + 2$ time units.*

Proof. In order to prove this theorem we should note that every cycle C_j can have at most 2 vertices that are at distance d from the originator. There can be any number of cycles that contain at most 2 vertices that are at distance d from the originator. Consider any cycle C_j that contains two vertices v_1 and v_2 that are at distance d from the originator. Fig. 10 shows vertices v_1 and v_2 that are on the cycle C_j. They are both at distance d from the originator (which is not shown in the graph). Note that both of these vertices are at equal distance, 7,

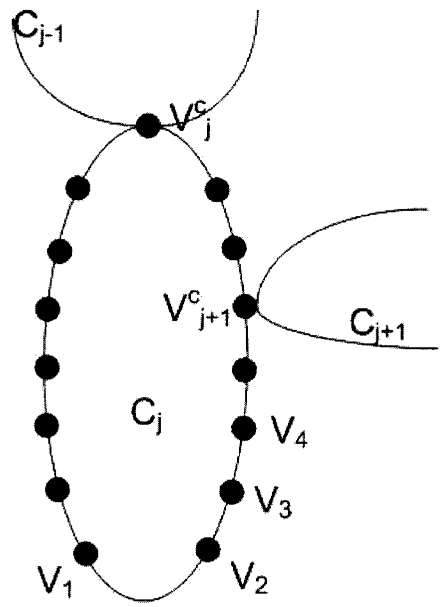

Figure 10: This figure shows a cycle in a necklace graph with 2 vertices which are at distance d (diameter) from the originator.

from the connection vertex v_j^C. According to the greedy scheme vertex v_{j+1}^C which happens to be a vertex on C_j and a connecting vertex connecting C_j to C_{j+1} will spend 2 time units informing 2 of its uninformed neighbors in C_{j+1}. Therefore, vertex v_1 will be informed at time d but v_2 will not be informed by this time because of the delay at the connecting vertex v_{j+1}^C. v_1 will inform v_2 at time $d+1$ and finally v_3 can be informed by either v_2 or v_4 at time $d+2$. Regardless of how many cycles have vertices at distance d from the originator, all the vertices of thoese cycles will be informed by the time $d+2$ with the greedy scheme. This concludes the proof. □

Finally we illustrate that $d+2$ is a tight upper bound by giving an example in Fig. 11. This example shows a necklace graph with 4 cycles where the last three cycles have each 2 vertices at distance d from the originator v_o. The vertices v_{20} and v_{21} in C_1 are at distance

14 from v_0, vertices v_{30} and $v31$ in C_2 are at distance 14 from the originator and the vertices v_{33} and v_{34} in C_3 are at distance 14 from the originator. The broadcast time of this graph is equal to the broadcast time of vertex v_0 and is equal to 16. Hence this is an illustration of a case where broadcast time in a necklace graph is $d + 2$.

4.2.6 2-Restricted Cactus Graphs

In this final subsection we will use the results presented in this chapter to present a partial solution of the broadcast problem in 2-restricted cactus graphs. Cactus graphs are defined to be connected graphs where no two cycles have more than one vertex in common. In a cactus graph it is possible to have a vertex that belongs to more than two cycles. We define the 2-restricted cactus graphs as graphs which are cactus graphs but there is no vertex that belongs to more than two cycles. A divide and conquer approach can be used as described in this chapter to solve the broadcast problem for all the originators that are not connecting two cycles.

Assume that the originator v_o in G belongs to a cycle C. As was done for tree of cycles, the graph G will be divided into smaller parts where each subgraph is in turn a 2-restricted cactus graph. The broadcast problem can be solved in each one of them and the proper broadcast trees are joined back to the cycle to solve the problem in the original graph. In order to prove this algorithm we need to prove the generalized version of Theorem 15.

In the following theorem we assume that graph G has a cycle C and graphs N_i , $1 \leq i \leq k$, attached to C each at vertex v_i. The originator is on the cycle C but is different than the vertices v_i. Furthermore, we assume that in each graph N_i the degree of v_i is 2, i.e.

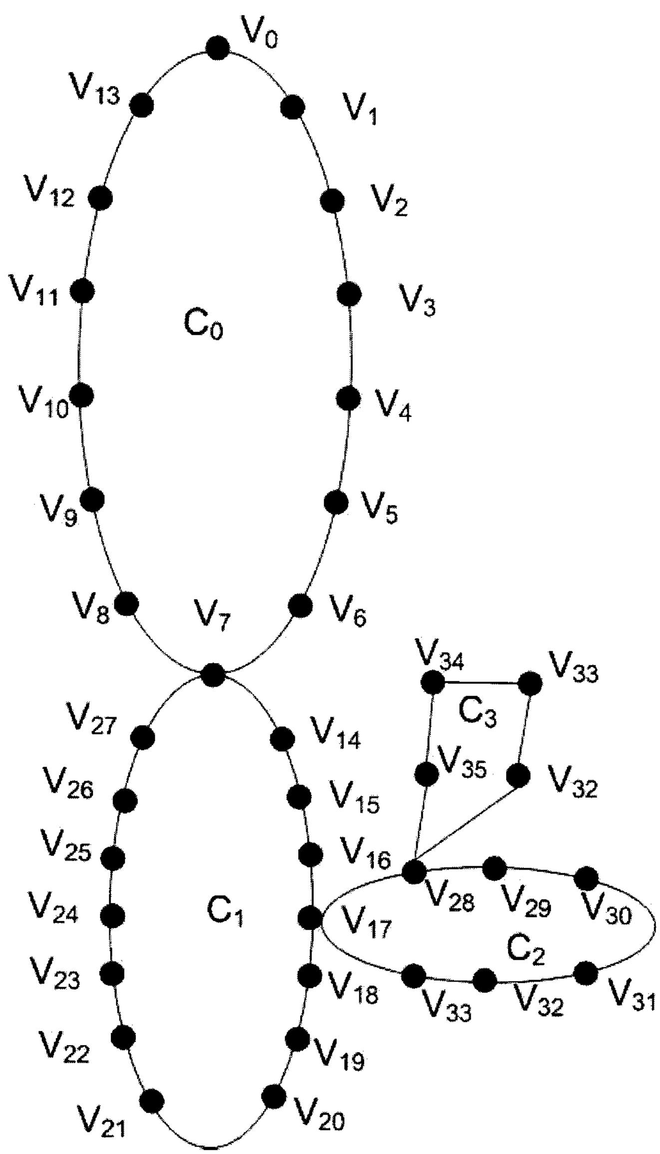

Figure 11: A necklace graph with 4 cycles. Cycles C_1 , C_2 and C_3 each have 2 vertices which are at distance d (diameter) from the originator v_o.

$\deg(v_i) = 2$. Then we can prove the following:

Theorem 19. *For the graph G consisting of the cycle C and graphs N_i as described above, there exists a broadcast tree T such that each subtree T_i induced by the vertices of N_i is a broadcast tree of N_i.*

Proof. The proof of Theorem 15 can be applied by simple generalization since in that proof every graph N_i, for $i = 1, 2$, was considered separately. The same procedure to replace T_i by a broadcast tree of N_i can be repeated k times to get an optimal broadcast tree. □

Moreover note that for each subgraph N_i in a 2-restricted cactus graph, the vertex v_i that connects N_i to the rest of the graph has degree 2 in N_i and a degree at most 4 in G. This implies that when multiple broadcast schemes exist in N_i each giving a different broadcast tree, then the tree that has the largest imbalance in the broadcast time between the two trees connected to vertex v_i is chosen to be attached to the rest of the graph.

Assume that the tree T_i' rooted at v_i is a broadcast tree of N_i for the originator v_i such that the difference between the broadcast time of its two children is the maximum among all broadcast trees of N_i.

Theorem 20. *For any originator not belonging to N_i, there exists a broadcast tree T in G such that each subtree induced by the vertices of N_i is equal to T_i'.*

If the the originator in a 2-restricted cactus graph, $G = (V, E)$, belongs to two cycles then special treatment might be needed as it was the case for necklace graphs. Assume that the originator v_o is the intersection of two cycles C_1 and C_2. The broadcast algorithm proceeds as follows. The graph G is divided into two parts $G_1 = (V_1, E_1)$ and

$G_2 = (V_2, E_2)$ such that $V_1 \cap V_2 = \{v_o\}$, $V_1 \cup V_2 = V$, $E_1 \cap E_2 = \emptyset$, and $E_1 \cup E_2 = E$. If $b(v_o, G_1) \neq b(v_o, G_2)$ then the optimal broadcast trees of G_1 and G_2 are used to calculate the broadcast time $b(v_o, G)$. However, if $b(v_o, G_1) \neq b(v_o, G_2)$ then in addition to calculating the broadcast trees of G_1 and G_2 we need to calculate the spanning tree T_1' and T_2' such that $b(v_o, T_1') = b(v_o, G_1) + 1$ and $b(v_o, T_2') = b(v_o, G_2) + 1$. Again among all the possible trees that can be obtained we need to choose the ones that have the maximum imbalance in the broadcast time between the two subtrees. However, we have no algorithm to calculate the trees T_1' and T_2'. Each tree T_i', $1 \leq i \leq 2$, is a spanning tree of a 2-restricted graph with broadcast time 1 more than the optimal broadcast time. The difficulty is in the fact that T_i' is calculated from a unicyclic graph where every tree attached to the cycle is the optimal solution of a subgraph. In the necklace graphs it was possible to obtain a T_i' because we showed that we only need to use either optimal solutions of the subgraphs or a spanning tree that has broadcast time one more than the optimal value. Also, in necklace graphs whenever the tree attached to the cycle was substituted with a one with a broadcast time larger by one time unit, we could always calculate in constant time the broadcast time of the resulting graph. However, in the general case of unicyclic graphs similar conclusions cannot be obtained easily. For example it is possible to substitute a tree T_i with another tree T_i' such that $b(r_i, T_i') = b(r_i, T_i) + 1$ but still obtain the optimal broadcast time in the unicyclic graph. The main question is the following: Given a graph G which has a cycle C and graphs N_i, $1 \leq i \leq k$, attached to the cycle at vertices r_i such that the degree of r_i in N_i is equal to 2, how can we construct all the spanning trees of G such that the broadcast time of an originator in these spanning trees is one more than the broadcast time of the graph

G. We showed above that we can construct all the broadcast trees of G. If we know how to find all these spanning trees in polynomial time then the broadcast problem in 2-cactus graphs can be solved.

Chapter 5

Fully Connected Trees

Assume that we have a complete graph where every vertex is the root of a tree. We will call the resulting graph fully connected trees. In this chapter we will study the broadcast problem in this kind of graph.

Definition 6. *Consider n trees $T_i = (V_i, E_i)$ rooted at r_i where $1 \leq i \leq n$. We define the fully connected tree, $FCT_n = (V, E)$, to be a graph where $V = V_1 \cup V_2 \cup \cdots \cup V_n$ and $E = E_1 \cup E_2 \cup \cdots \cup E_n \cup E_{K_n}$ where $E_{K_n} = \{(r_i, r_j) | 1 \leq i, j \leq n, i \neq j\}$. The roots of the trees, r_i, will be called root vertices and the rest of the vertices will be called tree vertices.*

We will first present a broadcast algorithm where the originator is a root vertex. A broadcast algorithm for the case the originator is not a root vertex will be presented in the second half of this chapter.

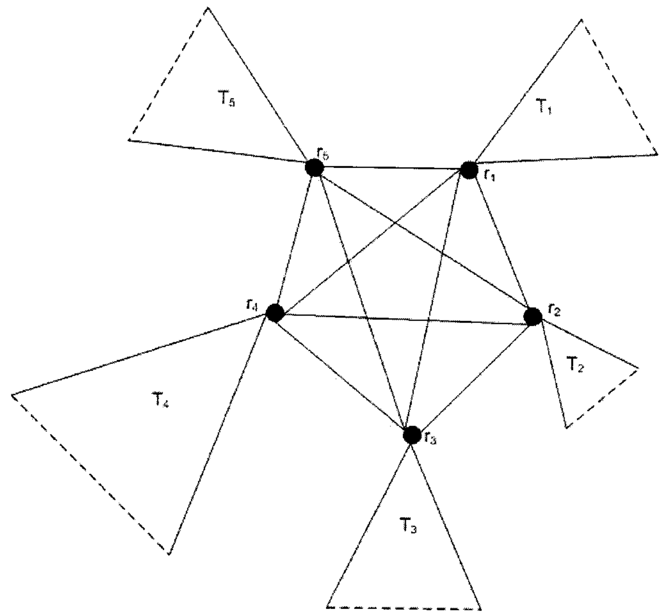

Figure 12: A fully connected tree FCT_5 with 5 trees T_i rooted at r_i, $1 \leq i \leq 5$. Note that the roots r_i induce a subgraph which is the complete graph K_5.

5.1 A Broadcast Algorithm for Root Vertices

In this section we consider the broadcast problem in fully connected trees. An upper bound on broadcast time can be obtained by a broadcast algorithm that first informs all the vertices of the complete graph and when all the root vertices are informed, each vertex informs the tree attached to it. In this case the time needed to inform all the vertices will be $t_{max} = \lceil \log n \rceil + \max\{b(v_i, T_i)\}$, where $b(v_i, T_i)$ is the broadcast time of the vertex v_i in the tree T_i. Similarly one can see that a lower bound on the broadcast time is $t_{min} = \max\{\lceil \log n \rceil, \max\{b(v_i, T_i)\}\}$.

5.1.1 The Broadcast Algorithm

First we will develop an algorithm that answers the following question: Given a time τ, a graph FCT_n, and an originator v_o, is it possible to complete broadcasting in FCT_n in at most τ time units? Obviously τ should be greater than or equal to t_{min}. Also, if τ is greater than t_{max} then the answer to the above question is trivially true. Below we will describe how to answer the question in the case $t_{min} \leq \tau \leq t_{max}$.

The algorithm takes the graph FCT_n, originator v_o which is a root vertex, and the candidate broadcast time τ as input parameters. The main idea of the algorithm is to do broadcasting assuming that the broadcast time is τ. If $\tau \geq b(v_o, FCT_n)$ then the algorithm will return TRUE and will inform all the vertices of the graph. Otherwise, it will return FALSE, meaning that τ is too small to be the broadcast time. If all the vertices of the graph are informed in τ time units or less, all we can conclude is that the broadcast time is less than or equal to τ. In order to conclude is that τ is the broadcast time we need to be able to inform all the vertices in τ time units and our algorithm should also return FALSE when $\tau - 1$ is given as input for the candidate broadcast time.

The first step of the algorithm will be to assign weights to every root vertex of the fully connected tree. The weight of each vertex r_i is initialized to the broadcast time $b(r_i, T_i)$. At every time unit t, where $0 \leq t \leq \tau$, each vertex will have to determine if τ is enough to broadcast the information to all the vertices. The algorithm terminates if a vertex can decide that τ will not be enough to inform all the vertices of the FCT_n. Otherwise, the algorithm continues and the informed vertices pass the message to uninformed neighbours. This process iterates until either all the vertices are informed or one of the informed vertices concludes that τ cannot be the broadcast time.

When a tree vertex is informed there is not much it can do other than following the well known broadcast algorithm in trees. However, when a root vertex is informed it has the option of passing the information to another root vertex or pass it down to the tree attached to it. Deciding on how every informed root vertex should choose an uninformed neighbour to pass the information to is the main step of the algorithm.

Algorithm 5.1: The decision algorithm to decide if a given time τ is enough to inform all the vertices of a graph FCT_n.

Algorithm: The Decision Algorithm: $DecisionAlgo(G, v, t)$
Input: $FCT_n = (V, E)$, originator v_o, candidate broadcast time τ
Output: FALSE if τ cannot be the broadcast time, TRUE if broadcasting can be accomplished in time less or equal to τ

```
Initialize V_I such that V_I = v_o ;
foreach t such that 0 ≤ t ≤ τ − 1 do
    foreach v ∈ V_I do
        if v is a root vertex then
            if w(v, t) > τ − t then
                v informs another uninformed root vertex at time t which has the
                highest weight among the uninformed root vertices ;
                Append the newly informed vertex to V_I ;
            else
                if w(v, t) = τ − t then
                    v informs one of its children which are tree vertices ;
                    Append the newly informed vertex to V_I ;
                else
                    return FALSE ;
                end
            end
        else
            v informs a tree vertex based on the tree broadcast algorithm in the
            uninformed subtree rooted at v ;
            Append the newly informed vertex to V_I ;
        end
    end
end
return TRUE ;
```

In order to make the description of the algorithm simpler each root vertex r_i will be assigned a weight $w(r_i, t)$. These weights will be equal to the broadcast time of r_i in the connected uninformed subtree of T_i rooted at r_i. In more details, the weight of each root vertex, r_i, at any time t will be calculated as follows: If r_i has no uninformed children which are tree vertices then its weight is zero because r_i can do nothing to speed up the process of informing the vertices of the subtree T_i. If r_i has one or more uninformed child, let the tree $T_{i,t}$ rooted at r_i be the sutbree of T_i which consists of only those vertices of T_i that are still not informed. The weight of r_i at time t, $w(r_i, t)$, will be equal to the broadcast time of r_i in the tree $T_{i,t}$, $b(r_i, T_{i,t})$. Next, we will describe how does an informed root vertex, r_i, at time t decide whether it has to inform another root vertex or a tree vertex. Two cases may arrise:

1. r_i does not have uninformed children which are tree vertices, in this case $w(r_i, t) = 0$. In this case there is nothing r_i can do to speed up the broadcast process in the tree attached to it. Therefore, r_i should inform an uninformed root vertex with the highest weight among the vertices.

2. r_i has uninformed children which are tree vertices. In this situation r_i has 2 options. One is to inform an uninformed root vertex and the other is to inform an uninformed neighbouring tree vertex. The choice between informing a root vertex versus a tree vertex is done by comparing the time needed to inform the uninformed subtree attached to it, $b(r_i, T_{i,t}) = w(r_i, t)$, with the remaining time $\tau - t$. If $\tau - t > w(r_i, t)$ then v informs another root vertex. If $\tau - t = w(r_i, t)$ then v has to inform one of its children i.e. a tree vertex according to the broadcast algorithm in trees. The case

where $\tau - t < w(r_i, t)$ is not being considered here because as soon as that happens we conclude τ cannot be the broadcast time and the algorithm return FALSE.

In Algorithm 5.1 we describe the decision algorithm that given a graph FCT_n, an originator v_o, and a candidate broadcast time τ, decides if the broadcast time of the graph is less than or equal that τ. If the candidate time τ happens to be the broadcast time, then this algorithm also generates the optimal broadcast scheme. Note that in the psuedocode V_I represents the set of informed vertices which at the start of the algorithm contains only v_o.

Now we are in a position to describe a broadcast algorithm for fully connected trees using the decision algorithm presented above. The algorithm does a binary search for the broadcast time in the range of possible values. As mentioned above we already know the minimum and maximum of the range which are t_{min} and t_{max}. The binary search reduces the size of the range by applying the decision algorithm on the midpoint of the range. If the algorithm returns that broadcasting can be performed, then the lower half of the range is considered for the recursive call of the search algorithm. However, if the result of the algorithm is negative, meaning that the midpoint of the range cannot be the broadcast time, then the upper half of the range is considered and the algorithm is again applied recursively. The psuedocode of the algorithm is presented in Algorithm 5.2. The initial call of the algorithm will be $BroadcastAlgorithmFCT_n(FCT_n, v_o, t_{min}, t_{max})$ where $t_{min} = \max\{\lceil \log n \rceil, \max b(v_i, T_i)\}$ and $t_{max} = \lceil \log n \rceil + \max\{b(v_i, T_i)\}$.

Finally in figure 13 we provide a simple example which depicts the different operations performed by the decision algorithm. We are given a graph which is a fully connected tree

Algorithm 5.2: The broadcast algorithm of an root vertex originator v_o in a graph FCT_n.

Algorithm: The Broadcast Algorithm $BroadcastAlgorithmFCT_n(G, v_o, t_1, t_2)$
Input: $FCT_n = (V, E)$, originator v_o, the minimum of a time range, and the maximum of a time range.
Output: Broadcast time τ such that $\tau = b(v_o, FCT_n)$

```
t = t1 + ⌊(t2 − t1)/2⌋ if t1 = t2 then
    if DecisionAlgo(G, vo, t1) then
        return t
    else
        return -1
    end
end
if t1 + 1 = t2 then
    if DecisionAlgo(G, vo, t1) AND !DecisionAlgo(G, vo, t2) then
        return t1
    end
    if !DecisionAlgo(G, vo, t1) AND DecisionAlgo(G, vo, t2) then
        return t2
    else
        return -1
    end
end
if DecisionAlgo(G, vo, t1) then
    return BroadcastAlgorithmFCTn(G, vo, t1, t)
else
    return BroadcastAlgorithmFCTn(G, vo, t + 1, t2)
end
```

with 5 trees, T_i where $1 \leq i \leq 5$. The originator is vertex r_1, the root of tree T_1, and the candiate broadcast time $\tau = 6$. The figure represents a snapshot of the state of the graph at time $t = 1$ where the informed vertices are r_1 and r_3. In order to decide which vertices should be informed next at time $t = 2$, the weights $w(v_i, t)$ should be calculated for $t = 1$. Since there are no tree vertices informed yet, the weights will be calculated as follows: $w(v_i, 1) = b(v_i, T_i)$. Hence, we obtain that $w(r_1, 1) = 1$, $w(r_2, 1) = 3$, $w(r_3, 1) = 5$, $w(r_4, 1) = 2$, and $w(r_5, 1) = 2$. The remaining time $t_{rem} = \tau - t$ is equal

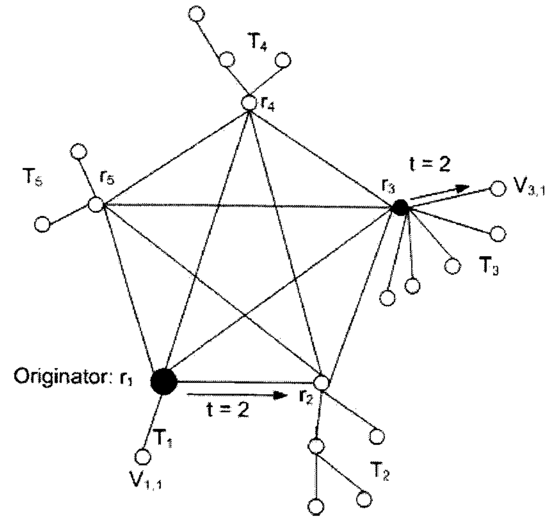

Figure 13: A fully connected tree FCT_5 with 5 trees T_i rooted at r_i, $1 \leq i \leq 5$. The originator is r_1 and at time $t = 1$ the informed vertices are r_1 and r_3. This figure shows that at time $t = 2$ the two vertices that get informed are r_2 and $v_{3,1}$.

to $t_{rem} = 6 - 1 = 5$. None of the informed vertices has a weight greater than 5 so the algorithm does not return FALSE. The informed vertices r_1 and r_3 inform new vertices at time $t = 2$ as follows: Vertex r_3 has weight $w(r_3, 1) = 5 = t_{rem}$, therefore it informs a tree vertex based on the well known tree algorithm. However, vertex r_1 informs another root vertex since $w(r_1, 1) = 1 < t_{rem} = 5$. It informs vertex r_2 since it has the greatest weight among the uninformed root vertices. The weights of the vertices $w(r_i, 2)$ remain the same for all vertices except r_3. Since r_3 has one child tree vertex that is informed, its weight $w(r_3, 2) = 4$.

5.1.2 Proof of Correctness

In this section we will prove the correctness of the broadcast algorithm presented in the previous section. The first algorithm, $DecisionAlgo(G, v, \tau)$ generates a broadcast scheme,

S_T, as well as a boolean value which indicates whether or not broadcasting is possible in the provided amount of time. At any time t any broadcast scheme S will have informed a certain number of vertices. We will denote the set of informed root vertices by any scheme S until time t by $V_t(S)$. First we will prove two results which will be used in verifying the correctness of the broadcast algorithm described above.

Assume we are given an infinitely large complete graph in which we study the broadcast process. Assume that at time t the number of informed vertices is N_t. We are required to calculate the number of informed vertices at time $t + \Delta$ with the following conditions:

1. Broadcasting is done according to the classical model.
2. There are α different vertices (out of N_t) that will stay idle for one time unit, in other words, the vertex will not inform a new vertex in the complete graph during that time unit.
3. The idle time unit can be anytime in the time interval $[t, t + \Delta]$ where Δ is any positive integer.

Lemma 6. *The number of informed vertices at time* $t + \Delta$ *will be maximum if all of the* α *vertices choose to remain idle at time* $t + \Delta - 1$.

Proof. First let us calculate the number of vertices when all of the α vertices remain idle at time $t + \Delta - 1$. It is clear that the number of vertices will double with every time unit when all of the vertices are informing a new vertex. Therefore, at time $t + \Delta - 1$ the number of informed vertices will be $|N_t|2^{\Delta-1}$. During the last time unit all but α of the informed vertices will inform a new vertex. Hence, the number of vertices will be

$|N_t|2^{\Delta-1}+(|N_t|2^{\Delta-1}-\alpha)$ which is equal to $|N_t|2^{\Delta}-\alpha$. Now lets consider the general case and calculate the number of informed vertices at time $t+\Delta$ assuming that the α vertices decide to stay idle at a time which is not necessarily at the last time unit. Consider a vertex v at time t, if v had an idle time unit δ units, $0 \leq \delta \leq \Delta - 1$, before the time $t+\Delta$, then the number of vertices will be $2^{\Delta} - 2^{\delta}$. Putting this together we can calcuate the total number of informed vertices at time $t+\Delta$ which is equal to $n_{max} = |N_t|2^{\Delta} - 2^{\delta_1} - \cdots - 2^{\delta_\alpha}$. This formula can be understood by noticing that $|N_t|2^{\Delta}$ is the maximum number of vertices that can be informed in Δ time units and we are subtracting the number of vertices that will not be informed whenever one of the α vertices stays idle δ_i time units before the end, $1 \leq i \leq \alpha$ and $0 \leq \delta_i \leq \Delta - 1$. Note that the maximum value of the formula n_{max} occurs when all of the negative terms are minimum, which implies that $\delta_i = 0$. Hence we proved that the maximum number of informed vertices at time $t+\Delta$ can be obtained when all of the α vertices stay idle at the last time unit and not before that. □

It is worth pointing out the relevance of this lemma to the problem of broadcasting in fully connected trees. The idle time unit of a vertex in the above lemma corresponds to a root vertex spending one time unit informing a tree vertex rather than informing another root vertex.

The time intervals discussed above are also relevant to the broadcast problem in fully connected trees. Consider a fully connected tree G with k root vertices r_i, $1 \leq i \leq k$. We will denote by δ_i the number of neighboring tree vertices that the vertex r_i has, in other words, $\delta_i = \deg(r_i) - k - 1$, where $\deg(r_i)$ is the degree of the vertex r_i. Assume that $\tau = b(v_o, G)$. The weight of each vertex r_i will have δ_i different values during the broadcast process. Every time r_i informs one of its tree vertex neighbors, its weight will decrease. The δ_i weights of r_i will be represented by $w_j(r_i)$ where $1 \leq j \leq \delta_i$. The number of these weights for all the vertices of the graph G is equal to $\sigma = \sum_{i=1}^{k} \delta_i$. Sort these weights in decreasing order and label them with with new varialbes ω_i where $1 \leq i \leq \sigma$, i.e. $\omega_1 \geq \omega_2 \geq \cdots \geq \omega_\sigma$. Now we can define the time intervals by considering the values $\tau_i = \tau - \omega_i$, $1 \leq i \leq \sigma$ and $\tau = b(v_o, G)$. Note that $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_\sigma$. Using these we can define σ time intervals as follows: $[0, \tau_1]$, and $[\tau_i + 1, \tau_{i+1}]$ where $1 \leq i \leq \sigma - 1$ which we denote by I_j where $1 \leq j \leq \sigma$. Some of these intervals can be overlapping since it is possible to have several vertices having the same weight. Note that during each time interval I_i, $1 \leq i \leq \sigma$, there is a vertex that has to spend one time unit informing a tree vertex. If there are p overlapping intervals with I_i then there are p vertices that have to inform a tree vertex during the time interval I_i.

Lemma 7. *Let G be a FCT such that $b(v_o, G) = \tau$. Let S_{opt} be an optimum broadcast scheme different than S_τ. Then, at any time τ_i, $|V_{\tau_i}(S_\tau)| \geq |V_{\tau_i}(S_{opt})|$ where $1 \leq i \leq \sigma$.*

Proof. We will prove this result by induction. For the base case, we can easily conclude that $|V_{\tau_1}(S_\tau)| \geq |V_{\tau_1}(S_{opt})|$ since at time $t = 0$ only the originator is informed in both schemes, hence $|V_0(S_\tau)| \geq |V_0(S_{opt})|$. Using the lemma proved above we can conclude the correctness of the base case. Now assume that $|V_{\tau_i}(S_\tau)| \geq |V_{\tau_i}(S_{opt})|$, we need to prove that $|V_{\tau_{i+1}}(S_\tau)| \geq |V_{\tau_{i+1}}(S_{opt})|$.

We can write $V_{\tau_i}(S_\tau) = V_c \cup V_n$ where V_c is the set of vertices that have to inform a tree vertex at time τ_i and V_n is the set of the rest of the vertices. Note that $V_c \subset V_{\tau_i}(S_{opt})$ otherwise S_{opt} cannot inform all the vertices of G in τ time units. Moreover, we can subdivide V_c as follows: $V_c = V_{cc} \cup V_{nc}$ where V_{cc} is the set of vertices that have to inform a tree vertex in the scheme S_{opt} and V_{nc} is the set of vertices that do not have to inform a tree vertex at time τ_i. With these we can calculate the number of vertices $|V_{\tau_i+1}(S_\tau)|$ and $|V_{\tau_i+1}(S_{opt})|$ as follows:

$$|V_{\tau_i+1}(S_\tau)| = 2(|V_{\tau_i}(S_\tau)| - |V_c|)$$

$$|V_{\tau_i+1}(S_{opt})| = 2(|V_{\tau_i}(S_{opt})| - |V_{cc}|)$$

. Moreover, we can conclude that there are $|V_{nc}|$ vertices in $V_{\tau_i}(S_{opt})$ that spent at least one time unit each informing a tree vertex, therefore $|V_{\tau_i}(S_\tau)| - |V_{\tau_i}(S_{opt})| \geq |V_{nc}|$. Putting all these together we can write:

$$|V_{\tau_i+1}(S_\tau)| - |V_{\tau_i+1}(S_{opt})| = 2(|V_{\tau_i}(S_\tau)| - |V_{\tau_i}(S_{opt})|) - 2(|V_c| - |V_{cc}|)$$

$$|V_{\tau_i+1}(S_\tau)| - |V_{\tau_i+1}(S_{opt})| = 2(|V_{\tau_i}(S_\tau)| - |V_{\tau_i}(S_{opt})|) - 2|V_{nc}|$$

Since $|V_{\tau_i}(S_\tau)| - |V_{\tau_i}(S_{opt})| \geq |V_{nc}|$, we conclude that $|V_{\tau_i+1}(S_\tau)| - |V_{\tau_i+1}(S_{opt})| \geq 0$ or $|V_{\tau_i+1}(S_\tau)| \geq |V_{\tau_i+1}(S_{opt})|$. Finally, together with the above lemma we can conclude that $|V_{\tau_{i+1}}(S_\tau)| \geq |V_{\tau_{i+1}}(S_{opt})|$. □

Lemma 8. *Let G be a FCT such that $b(v_o, G) = \tau$. Let S_{opt} be an optimum broadcast scheme different than S_τ. Then, at any time t we have $|V_t(S_\tau)| \geq |V_t(S_{opt})|$.*

Proof. First note that time t falls in one of the time intervals as defined above, without any loss of generalization assume that t is in the time interval $[\tau_i + 1, \tau_{i+1}]$. Because of the previous lemma we know that $|V_{\tau_{i+1}}(S_\tau)| \geq |V_{\tau_{i+1}}(S_{opt})|$ and because of lemma 6 we can deduce that at any time t in this interval $|V_t(S_\tau)| \geq |V_t(S_{opt})|$. □

Theorem 21. *Given a graph G, an originator v_o, and a time τ, if $DecisionAlgo(G, v_o, \tau)$ returns false then $b(v_o, G) > \tau$.*

Proof. Assume that there exists a scheme S such that all the vertices of G are informed within τ time units or less. Since $DecisionAlgo(G, v_o, \tau)$ returned false, there was a root vertex v which got informed at time t and $w(v, t) > t - \tau$. 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 in the scheme S_τ, $V_{t-1}(S_\tau)$, all have weights greater than or equal to $w(v, t)$. Since S is a broadcast scheme then 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 v at time $t-1$ or earlier, since it has a weight greater than $\tau - t$. 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 the previous lemma. Therefore, we conclude that such a scheme S cannot exist if the $DecisionAlgo(G, v_o, \tau)$ returns false. □

5.1.3 Complexity Analysis

In this section we will calculate the complexity of the algorithm described above. First one can note that the $BroadcastAlgorithm(G, v, t_1, t_2)$ does a binary search for the broadcast time in the range of possible values. 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. However, every time we need to verify if a certain value in the range is less than, greater than, or equal to what we are looking for, we are running the $DecisionAlgo(G, v, t)$ which has a linear complexity in the number of vertices of the graph G. Assume that n is the number of vertices of the graph G. The $DecisionAlgo(G, v, t)$ can calculate its decision in a linear time because every root vertex has to calculate its weight and compare 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 of the graph because the tree broadcast algorithm has to run which is linear itself. Therefore, the complexity of the $BroadcastAlgorithm(G, v, t_1, t_2)$ has a complexity of $O(n \log(t_2 - t_1))$. Also note that $t_1 - t_2 = O(\log n)$ which implies that the complexity of the algorithm is $O(n \log \log n)$.

The final step is to show that the decision algorithm has a complexity linear in n. The $DecisionAlgo(G, v, t)$ can calculate its decision in a linear time because every root vertex has to calculate its weight and compare 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 that will be needed before one of the root vertices makes a decision is at most equal to the total number of children vertices that the root vertices have. This number can be a linear function of the total number of vertices.

5.2 Broadcasting from any originator

In the previous sections we assumed that the originator is always one of the root vertices. In this section we will develop a broadcast algorithm for any originator in an arbitrary fully connected tree FCT. Assume we are given a fully connected tree G such that the originator v_o is in the tree T_i rooted at one of the root vertices r_i. There is a unique path P in T_i connecting r_i to the originator v_o. The vertex on the path P neighbouring r_i will be denoted by v_i. Let u_j, $1 \leq j \leq k$, be the neighbors of v_o in the tree. One of these vertices falls on the path P, call this vertex u_i. The subtree of T_i rooted at the vertex v_i will be called T(see Fig 14a). The remaining subtree of T_i, rooted at r_i, after removing the edge (r_i, v_i) will be called T_i'. We will construct a new graph G' which is again a fully connected tree but the tree T_i' is attached at the root vertex r_i instead of the tree T_i. Figure 14a shows all the details described above. It is worth noticing that one can redraw the graph G differently by drawing the tree T rooted at the originator v_o and vertex v_i as one of its leaf vertices, this is shown in figure 14b. It can be observed that the graph G' is attached to the tree T by a bridge (v_i, r_i). Since graph G' is connected to the tree T by a bridge, the broadcast algirthm in G' is independent of the broadcast algorithm in T. Once vertex r_i is informed, it can not inform any other vertex in T so its job is to inform the vertices of G' in the fastest possible way. However, since G' is a fully connected tree and r_i is a root vertex, we have a broadcast algorithm to solve the broadcast problem of vertex r_i in G'.

Using Theorem 12 we can use the optimal broadcast tree of G' and attach it to the tree T and solve the broadcast problem in the resulting tree. In more details, according to the broadcast algorithm in trees [112], vertex v_o informs a child vertex that has the highest broadcast time in the subtree rooted at it. The subtrees rooted at the children of v_o are labeled by H_j ,$1 \leq j \leq k$, as shown in figure 14b. The broadcast times $b(u_j, H_j)$, $1 \leq j \leq k$, can be easily calculated except for the case where $u_j = u_i$. This is the case where there is the graph G' attached to v_i which might change the time needed by u_i to inform all of the vertices of H_i and G'. Since G' is a fully connected tree we can solve the broadcast problem for the originator r_i and obtain a broadcast tree $T_{G'}$. We construct a tree H_i' by attaching the tree H_i to $T_{G'}$ by the bridge (v_i, r_i). Using Theorem 12, the optimal time needed for u_i to inform all the vertices of H_i and G' is equal to the broadcast time of the vertex u_i in the tree H_u'.

In conclusion, finding the broadcast time of a tree vertex in an arbitrary fully connected tree can be reduced to solving two problems: one is finding the broadcast time of a root vertex in a fully connected tree and the second one is finding the broadcast time of a vertex in a tree. The complexity of the problem still remains $O(n \log \log n)$ because finding the broadcast time of a vertex in a tree is linear and hence the complexity is determined by the algorithm that calculates the broadcast time of a root vertex in an arbitrary FCT.

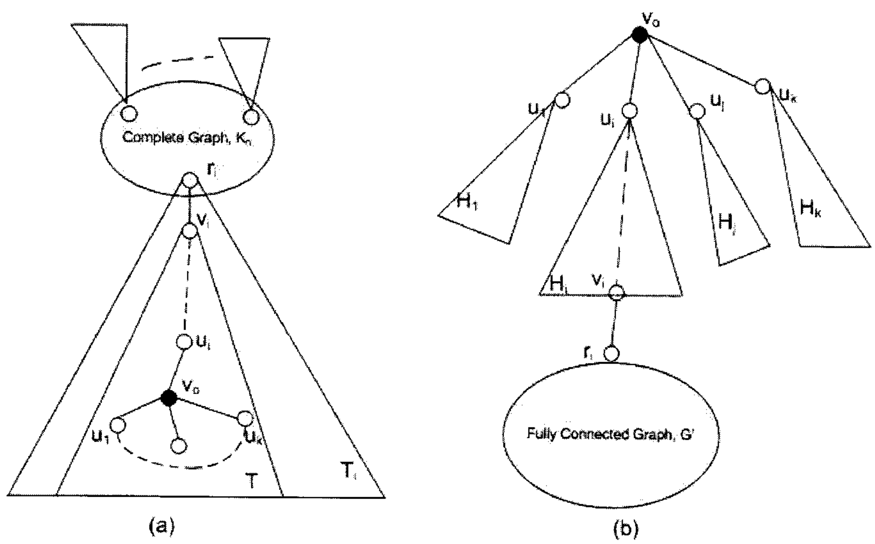

Figure 14: A fully connected tree G with an originator which is not a root vertex. (a) shows graph G and one of its trees T_i which contains the originator. (b) A different way of drawing the graph G by separating the tree T and the fully connected tree G'.

Chapter 6

Hierarchical Tree Cluster Networks

6.1 Introduction and Motivation

An interconnection network does not necessarily have the same performance requirements across the whole network. It often happens that physically close nodes on a network need to communicate more often with each other than those nodes that are far from each other. For such cases, it is meaningful to have a network design where the interconnection topology between physically close nodes is different than the topology connecting distant nodes. All nodes that are considered to be physically close will be considered to be a cluster. The nodes inside a cluster can have various interconnection topologies designed to meet the communication requirements of that cluster.

In this chapter we will study the message broadcasting problem in a network of clusters where the different clusters are connected to each other by a tree topology. We will consider various network topologies for the clusters and present an exact or an approximation algorithm for each case. We will refer to this kind of networks as hierarchical tree cluster networks (HTCN).

A hierarchical tree cluster network (HTCN) is a graph $G = (V, E)$ where the set of vertices can be divided into mutually disjoint sets V_i ,$1 \leq i \leq q$. The subgraphs induced by the sets V_i are called clusters. The different clusters are connected together by edges such that there is no cycle in the graph that contains vertices from two different clusters.

Definition 7. *A cluster $C_i = (V_i, E_i)$ is a graph which has the following properties:*

1. *There exists a subgraph of C_i which is a clique and is called the core of the cluster.*
2. *Every vertex in the core can perfrom optimum broadcasting in the cluster.*
3. *Only core vertices of a cluster can have connections to other clusters.*

Definition 8. *Consider m clusters $C_i = (V_i, E_i)$ $1 \leq i \leq m$ such that the sets V_i are mutually disjoint, the hierarchical tree cluster network , $HTCN$, is a connected graph $HTCN = (V, E)$ such that $V = V_1 \cup \cdots \cup V_m$ and $E = E_1 \cup \cdots \cup E_m \cup E_T$, where E_T is called the set of tree edges and $|E_T| = m - 1$. Each edge $e \in E_T$ connects vertices from two different clusters.*

We believe HTCNs can simulate more realistically networks that network engineers might face. According to the definition, in an HTCN not all nodes of a cluster can communicate with the other clusters directly, there is a set of nodes that are designated to have connections to other clusters. This set forms a clique and is called to be the core of the cluster. Having this kind of special nodes can be motivated by several reasons such as easier implementation of security protocols, and lower hardware cost. All the other nodes in the cluster are connected by some path to at least one designated node (a node in the core). Any such node willing to communicate with another node in another cluster does so by going through one of the designated nodes in its cluster which communicates with a designated node in the destination cluster which in turn knows how to inform any node in that cluster. Moreover, we assume that the interconnection topology inside a cluster is such that we know how to do optimal broadcasting in the cluster from any node in the core. Not all vertices know how to do optimal broadcasting in the cluster but they have routing tables which allow them to communicate with the closest designated node in the cluster which can do optimal broadcasting in that cluster. The different clusters are connected together with edges which connect a designated node of one cluster to another designated node of the other cluster. The connections between clusters is such that the graph reduces to a tree if every cluster was represented by only one vertex.

Note that, the subgraph induced by all the designated vertices of a hierarchical tree cluster graph is a tree of cliques. This observation will be useful in designing an approximation algorithm for broadcasting in HTCNs. A tree of cliques, TC, is an HTCN where every cluster forms a clique. TC is a graph which reduces to a tree if every clique was represented by only one vertex (Fig 15a).

Definition 9. *Consider m cliques $C_i = (V_i, E_i)$ $1 \leq i \leq m$ with sizes greater than or equal to 1. The tree of cliques, TC, is a connected graph $TC = (V, E)$ such that $V = V_1 \cup \cdots \cup V_m$ and $E = E_1 \cup \cdots \cup E_m \cup E_T$, where E_T is called the set of tree edges and $|E_T| = m - 1$. Each edge $e \in E_T$ connects vertices from two different cliques.*

To avoid confusion we will not call a graph HTCN if it is a tree of cliques even though by definition the set of graphs which are HTCN is a superset of the family of TCs. In this chapter we present a $O(n \log \log n)$ algorithm to determine the broadcast time of any vertex in an arbitrary TC. We use this algorithm to present a 3-approximation algorithm to find the broadcast time of any vertex in an arbitrary HTCN. This algorithm stays a constant approximation algorithm in the case where only a constant approximation algorithm is known to broadcast from the designated vertices of the clusters.

6.2 Broadcasting in Tree of Cliques

In this section we will study the broadcast problem in tree of cliques. We will describe a recursive divide and conquer broadcast algorithm for the tree of cliques G. For any graph $G = (V, E)$ the set of edges E can be divided into 2 disjoint sets. $E = E_c \cup E_t$ where the E_c are the set of clique edges, i.e. edges that connect the different vertices within each clique C_i. E_t is the set of edges such that each edge $(u, v) \in E_t$ connects two different cliques $C_i = (V_i, E_i)$ and $C_j = (C_j, E_j)$, $i \neq j$ and $1 \leq i, j \leq m$, such that $u \in V_i$ and $v \in V_j$. The recursive broadcast algorithm proceeds by choosing a clique C_i, $1 \leq i \leq k$, and removing the edges of E_t that connect C_i to neighboring cliques C_j. If there are k such edges then we will have $k+1$ disconnected graphs. One of them is the clique C_i and the rest of the graphs, which we denote by H_i where $1 \leq i \leq k$, are subgraphs of G such that each one of them in turn is a tree of cliques. The algorithm recursively calculates the broadcast tree of each graph H_i, $1 \leq i \leq k$, and then joins the results to construct a new graph, G', which is a fully connected tree. The construction of G' is done by using the broadcast tree, T_{H_i}, of each H_i graph. Each of these k trees are connected to G_i by putting back the k edges that were removed earlier. The last step will be to run the broadcast algorithm for fully connected trees in the graph G'. The pseudocode for the broadcast algorithm in TCs is given in Algorithm 6.1.

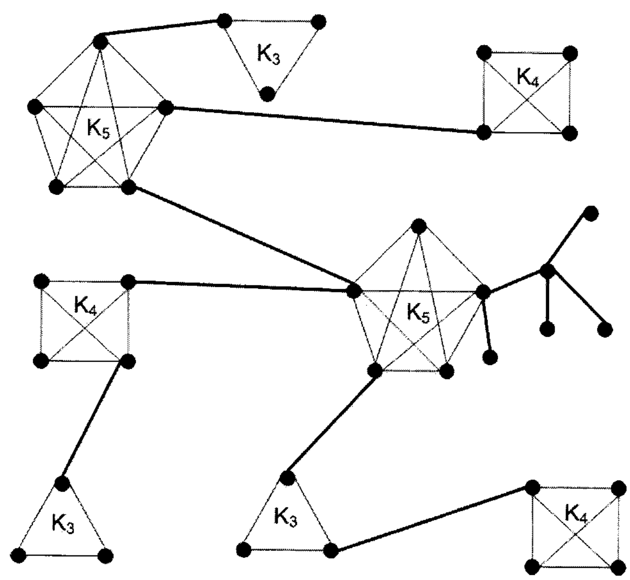

Figure 15: A tree of cliques with 13 cliques and 12 edges connecting those cliques (the tree edges are in bold and there are 5 cliques of size 1.)

6.2.1 Proof of Correctness

In order to prove the correctness of $BroadcastTC(TC, v_o)$ (Algorithm 6.1) we need to use Theorem 12 which was proved in Chapter 4. Note that the edges E_t of the graph are actually bridges since removing any of them results in a disconnected graph. Therefore, the broadcast algorithm correctly calculates the broadcast time of the graph G provided that the algorithm for calculating the broadcast time of fully connected trees is correct. The

Algorithm 6.1: The broadcast algorithm that calculates the broadcast time $b(v_o, G)$ and scheme for any originator v_o in any tree of cliques FCT.

Algorithm: $BroadcastTC(TC, v_o)$
Input: $TC = (V, E)$ and originator v_o
Output: Broadcast time $b(v_o, G)$ and the broadcast tree of G
Find the clique C_o such that the originator v_o belongs to it ;
Calculate the set of edges E_{C_o} which connect C_o to another clique ;
if E_{C_o} *is empty* **then**
    Calculate the broadcast time and the broadcast tree of the clique C_o.;
    return
else
    $G' \leftarrow C_o$;
    foreach $(v_i, u_j) \in E_{C_o}$ **do**
        remove edge (v_i, u_j) and obtain the connected subgraph H_j ;
        call $BroadcastTC(H_j, u_j)$ to obtain the broadcast tree T_{H_j} ;
        Add the edge (v_i, u_j) to the graph G' ;
        Add T_{H_j} to G' ;
    end
    Use $BroadcastFCT(G', v_o)$ to calculate the broadcast time and the broadcast scheme of the fully connected tree graph G' ;
end

correctness of the latter algorithm has been proved in the previous subsection.

6.2.2 Complexity Analysis

Assume we are given a tree of cliques G and that the originator is in clique C_o and there are q cliques in G. The first step is to divide the problem into smaller pieces. This is done by removing those edges that connect the clique C_o to other cliques. We will label these edges by E_o. Assume that there is m smaller graphs H_i which resulted after removing the tree edges. The next step is to solve the broadcast problem in each graph H_i which is itself a tree of cycles. The final step is to join the solutions by making use of the broadcast algorithm for fully connected trees. This joining step will take $O(k_o) + O(\log(t_{max} - t_{min}))O(n_o)$, where k_o is the number of vertices in the clique C_o and n_o is the number of children vertices that the root vertices of C_o have. Let U_i represent the time complexity of the broadcast problem in H_i. Therefore, the complexity of the algorithm is $U_1 + U_2 + \cdots + U_m + O(k_o) + O(n_o \log(t_{max} - t_{min}))$. The terms U_i can be each expanded and the resulting formula would be $\sum_{i=1}^{q} O(k_i) + O(\log(t_{max} - t_{min})) \sum_{i=1}^{q} O(n_i)$ where q is the total number of cliques, k_i is the number of vertices that the clique C_i has, and n_i is the number of children the root vertices of C_i have. The last step is to write the complexity in terms of n, the total number of vertices in the graph. Since $\sum_{i=1}^{q} O(k_i) < n$ and $\sum_{i=1}^{q} O(n_i) < O(n)$ we can write the complexity of the algorithm as $O(n) + O(n \log(t_{max} - t_{min}))$. Noting that $t_{max} - t_{min} = O(\log n)$, we can conclude that the complexity of the broadcast algorithm for tree of cycles in $O(n \log \log n)$.

6.3 An Approximation Algorithm for Hierarchical Tree Cluster Networks

In this section we will present an approximation algorithm for calculating the broadcast time of hierarchical tree cluster networks (HTCN).

Assume that the originator v_o is in the cluster C_j. The algorithm, A, we propose is hierarchical too in the sense that it first does broadcasting in the subgraph (induced by the cliques) which connects the different clusters together. In more details assume that v_o is not a designated node in the cluster and the closest designated node that v_o can reach is u. We denote by TC_G the tree of cliques subgraph of G which contains all the designated vertices of all clusters. The approximation broadcast algorithm first sends the message from v_o to u. Then broadcasting is performed in TC_G from the originator u. By the time the broadcast process in TC_G is complete, all the core vertices of the clusters have been informed. Then these vertices start broadcasting in their clusters. We are assuming that the clusters have such a network topology that we know how to do optimal broadcasting from any designated node in that cluster. Next we will show that this algorithm is a constant approximation algorithm.

Lemma 9. *Given a graph $G = (V, E)$, for any pair of vertices $x, y \in V$, $b(x, G) \leq 2b(y, G)$*

Proof. If $b(x, G)$ is already less than or equal to $b(y, G)$ then the result $b(x, G) \leq 2b(y, G)$ is trivially satisfied. Let us consider the case where $b(x, G) > b(x, G)$. Then we can write that, $b(x, G) \leq d(x, y) + b(y, G)$. Also we can deduce that $d(x, y)$ cannot be larger than $b(y, G)$ because the broadcast algorithm from the originator y can inform all the vertices of G including x within $b(y, G)$ time units, hence $d(x, y) \leq b(y, G)$. Putting these together we deduce that $b(x, G) \leq 2b(y, G)$. □

Theorem 22. *The broadcast time for the graph G calculated by algorithm A is at most three times the optimal broadcast time of G, i.e. $b_A(G) \leq 3b(G)$.*

Proof. Assume that the clusters are labeled such that $b(C_1) \geq b(C_2) \geq \cdots \geq b(C_q)$. Note that $b(C_i)$ is the maximum broadcast time among the designated nodes of the cluster and not the usual maximum broadcast time of the whole cluster (since we do not assume that we know how to perform broadcasting from non-designated nodes). Therefore we can write that $b_A(v_o, G) \leq d(v_o, u) + b(TC_G) + b(C_1)$ which directly follows from the definition of the algorithm. Once all the designated vertices of the clusters have been informed waiting $b(C_1)$ time units guarantees that all the vertices of the clusters have been informed because $b(C_1)$ is the largest value among all the clusters.

On the other hand, we know that $b(v_o, G) \geq d(v_o, u) + b(TC_G)$ since the broadcast time of a graph is always greater or equal than the broadcast time of any of its subgraphs.

Now we have two cases to consider:

1. v_o belongs to cluster C_1. Then using lemma 9, we can write that $b(v_o, G) \geq \frac{b(C_1)}{2}$.

2. v_o is not in cluster C_1. Then $b(v_o, G) \geq b(C_1)$ which trivially implies that $b(v_o, G) \geq \frac{b(C_1)}{2}$.

Therefore we conclude that $b(TC_G) + d(v_o, u) + \frac{b(C_1)}{2} \leq 2b(v_o, G)$. Substituting this in $b_A(v_o, G) \leq d(v_o, u) + b(TC_G) + b(C_1)$ we obtain that $b_A(v_o, G) \leq 2b(v_o, G) + \frac{b(C_1)}{2}$. Again using the fact that $b(v_o, G) \geq \frac{b(C_1)}{2}$ we obtain that $b_A(v_o, G) \leq 3b(G)$. Hence we proved that the described approximation algorithm is 3-approximation algorithm. □

Assuming that the complexity of the broadcast algorithm for a core vertex inside a cluster is $O(f(n))$, the complexity of the approximation algorithm is $O(n \log \log n + f(n))$. The $O(n \log \log n)$ factor is due to the fact that the approximation algorithm first does broadcasting in the subgraph of HTCN which is a tree of cliques. The $O(f(n))$ factor is contributed by the broadcast algorithm that broadcasts the information from the informed core vertices in each cluster. In practice a cluster will have a core of tightly connected vertices and the rest of the vertices will be weakly connected most probably with a tree structure. For such topologies, a linear time broadcast algorithm most probably exists. This implies that the complexity of the approximation algorithm that we presented is $O(n \log \log n)$ for a wide range of cluster topologies that one could face in real life.

It is interesting to see that the HTCN degenerates to a TC if some constraints are added. If every vertex, which is not a designated special vertex, has a unique path to one and only one designated vertex in the cluster, the resulting interconnection topology of the cluster will be a fully connected tree. In this case the algorithm for TCs presented in the previous section can calculate the exact broadcast time of the HTCN. This is because every clique will have the fully connected tree graph structure and the recursive divide and conquer algorithm of tree of cliques can be applied to obtain an exact $O(n \log \log n)$ broadcast algorithm.

Finally, we can extend the analysis to the approximation algorithm to cases when the exact broadcast algorithm from the designated vertices of a cluster is not known. Instead assume that a c-approximation algorithm is known for every designated vertex in a cluster. Therefore, for the same approximation algorithm A, can write that $b_A(v_o, G) \leq d(v_o, u) + b(TC_G) + cb(C_1)$. The upper bound on $b(v_o, G)$ is still the same namely, $2b(v_o, G) \geq b(TC_G) + d(v_o, u) + \frac{b(C_1)}{2}$. Using these two inequalities we can write that $b_A(v_o, G) \leq 2b(v_o, G) + \frac{2c-1}{2}b(C_1)$ which together with the inequality $b(v_o, G) \geq \frac{b(C_1)}{2}$ gives $b_A(v_o, G) \leq (2c+1)b(v_o, G)$. Hence, we conclude that our approximation algorithm is still a constant $(2c+1)$-approximation.

Chapter 7

Conclusion and Future Work

In today's world, interconnection networks are used to exchange information in many applications. Parallel computing is one example where fast message dissemination is of particular importance. Broadcasting is one of the message dissemination primitives that gets used frequently. In this thesis we studied the broadcast problem in interconnection networks focusing on several classes of networks.

The broadcast problem in general graphs is known to be NP-complete. Polynomial time solutions are known to exist only for a few classes of graphs such as the tree, hypercube, Knodel graphs, and the grid. The simplest graph structure among these is the tree while the rest of classes have properties such as regularity and symmetry that makes the broadcast problem solvable. Our choice of graph classes was motivated by the desire to increase the complexity of the graphs starting with the trees and adding more edges and gradually introducing cycles in the graph. Another direction in designing graphs was having the tree structure somehow hidden in the graph in the form of a backbone structure. In that respect we first studied the broadcast problem in unicyclic graphs where we provided a linear broadcast algorithm. The next step was to introduce cycles in a controlled way and we studied the tree of cycles, necklace graphs, and the 2-restricted cactus graphs. In all these graphs the presence of the tree structure was the ultimate reason a solution was obtained. In the second part of the thesis we studied graph structures where the main characteristic was the presence of a backbone tree network which connects different graphs together. In this category of graphs we considered the fully connected trees, the tree of cliques, and the hierarchical tree cluster networks. For the latter class we provide a constant approximation algorithm and for the rest we provide exact solutions.

There are other graph structures that might have polynomial solutions for the broadcast problem and have tree-like properties such as the partial k-trees. It is natural to study the broadcast problems in these kinds of graphs too. Partial k-trees are graphs which have constant treewidth and it is worth studying what classes of graphs with constant treewidth can be solved polynomially or at least constant approximation algorithms can be found. Another direction is to study the multicast problem in the graph families that we considered in this thesis. Multicasting is similar to the broadcast process with the difference that a message has to be sent only to a subset of destination vertices.

As stated in Chapter 2, another area of research has been designing networks with the least broadcast time possible given the number of vertices and edges. Formally speaking, the (n, m) problem is to design a graph $G = (V, E)$ such that $|V| = n$, $|E| = m$ and $b(G) = b_{min}$ such that there is no other graph G' on n vertices and m edges such that $b(G') < b(G)$. The only known results in this area is the solution of the $(n, n-1)$ problem (which are optimal trees) for the classical broadcast model, k-broadcast model, and universal list broadcast model. Given the extensive studies we have done in unicyclic graphs we plan to work on the (n, n) broadcast problem.

Bibliography

[1] R. Ahlswede, L. Gargano, H. Haroutunian and L. Khachatrian. Fault tolerant minimum broadcast networks, *Networks*, 27:293-307, 1996.

[2] R. Ahlswede, H. Haroutunian, and L. Khachatrian. Messy broadcasting in networks, *Communications and Cryptography*, eds. R.E. Blahut, D.J. Costello Jr., U. Mauter, and T. Mittelholzer (Kluwer, Boston/Dordrecht/London), pp. 13-24, 1994.

[3] W. Aiello, F. Chung, L. Lu, Random evolution in massive graphs, *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, FOCS'01*, pp. 510-519, 2001.

[4] S. Arnborg, J. Lagergren, D. Seese, Easy Problems for Tree-Decomposable Graphs, *Journal of Algorithms*, 12, pp 308–340, 1991.

[5] A. Bar-Noy, S. Guha, J. Naor and B. Schieber. Multicasting in Heterogeneous Networks, *Proc. of ACM Symp. on Theory of Computing, STOC'98*, 1998.

[6] A. Bar-Noy, S. Guha. Message multicasting in heterogeneous networks, *SIAM J. Comput.*, 30(2):347-358, 2000.

[7] A. Bar-Noy and S. Kipnis. Broadcasting multiple messages in simultaneous

send/receive systems, in *5th IEEE Symp. Parallel, Distributed Processing*, pp. 344-347, 1993.

[8] A. Bar-Noy, S. Kipnis, and B. Schieber. Optimal multiple message broadcasting in telephone-like communication systems, *Proc. Sixth Symp. Parallel and Distributed Processing*, pp. 216-223, 1994.

[9] B. Beauquier, S. Perennes, and O. Delmas. Tight Bounds for broadcasting in the linear cost model, *Journal of Interconnection Networks*, 2(2):175-188, 2001

[10] R. Beier, J. F. Sibeyn, A powerful heuristic for telephone gossiping, *Proc. of the 7th International Colloquium on Structural Information & Communication Complexity, SIROCCO'00*, L'Aquila, Italy, pp. 17–36, 2000.

[11] J-C Bermond and C. Peyrat, Broadcasting in deBruijn Networks, *Proc.* 19^th *South-eastern Conference on Combinatorics*, Graph Theory and Computing, Congressus Numerantium, 66:283–292, 1988.

[12] J. -C. Bermond, P. Fraigniaud, J. Peters, Antepenultimate broadcasting, *Networks*. 26, pp. 125–137, 1995.

[13] J. -C. Bermond, H. A. Harutyunyan, A. L. Liestman, and S. Perennes. A note on the dimensionality of modified Knödel graphs, *Int. J. Found. Comp. Sci.* 8, pp. 109–117, 1997.

[14] J. -C. Bermond, P. Hell, A. L. Liestman, J. G. Peters, Sparse broadcast graphs, *Discrete Appl. Math.* 36, pp. 97–130, 1992.

[15] J. -C. Bermond, P. Hell, A. L. Liestman, and J. G. Peters, Broadcasting in bounded degree graphs. *SIAM J. Discr. Math.* 5:10–24, 1992.

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

[17] H.L. Bodlaender. A tourist guide through treewidth, *Acta Cybernet*, 11, pp.1–21, 1993.

[18] S.C. Chau and A.L. Liestman. Constructing minimal broadcast networks, *J. Combin. Inform. System. Sci.*, 10:110-122, 1985.

[19] X. Chen. An upper bound for the broadcast function B(n), *Chinese J. Comput.*, 13:605-611, 1990.

[20] P. Chinn, S. Hedetniemi, and S. Mitchell. Multiple message broadcasting in complete graphs, in *Proc. Tenth Southeastern Conf. on Combinatorics, Graph Theory and Computing, Utilitas Mathematica*, Winnipeg, pp. 251-260, 1979.

[21] E. Cockayne and A. Thomason. Optimal Multi-Message Broadcasting in Complete Graphs, in *Proc. of 11th SE Conf. Combinatorics, Graph Theory, and Computing*, pp. 181-199, 1980.

[22] F. Comellas and C. Dalfo, Optimal broadcasting in 2-dimensional manhattan street networks, *Parallel and Distributed Computing and Networks*, 246:135-140, 2005.

[23] F. Comellas, H. A. Harutyunyan, and A. L. Liestman. Messy broadcasting in mesh and torus networks, *Journal of Interconnection Networks*, 4:37-51, 2003.

[24] F. Comellas, P. Hell. Broadcasting in generalized chordal rings, *Networks*, 42(3):123-134, 2003.

[25] A. L. H. Chow, L. Golubchik, S. Khuller, Y. Yao. On the tradeoff between playback delay and buffer space in streaming. *Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing IPDPS'09*, pp. 1–12, 2009.

[26] K. Diks and A. Pelc. Broadcasting with universal lists, *Networks*, 27(3):183-196, 1996.

[27] H. Dinh, Y.-A. Kim. On the Tree-Based Broadcast Algorithm (TBA) and an Improved Upper Bound on the Optimal Broadcasting/Gossiping Time in d-dimensional Torus Graphs. *Networks*, to appear, 2010.

[28] M. J. Dinneen. The complexity of broadcasting in bounded-degree networks, Tech. Report LACES-05C94 -31, Los Alamos National Laboratory, 1994

[29] M. J. Dinneen, M. R. Fellows, V. Faber, Algebraic constructions of efficient broadcast networks, *Applied Algebra, Algebraic Algorithms and Error- Correcting Codes* 9, Lecture Notes in Computer Science, vol. 539, Springer, Berlin, pp. 152–158, 1991.

[30] M. J. Dinneen, J. A. Ventura, M. C. Wilson, G. Zakeri, Compound constructions of broadcast networks, *Discrete Math.* 93, pp. 205–232, 1999.

[31] M. B. Doar, A better model for generating test networks, *IEEE GLOBECOM'96*, London, UK, 1996.

[32] M. Elkin, G. Kortsarz. A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem, *Proc. of ACM Symp. on Theory of Computing, STOC'02*, pp. 438–447, 2002

[33] M. Elkin and G. Kortsarz, Sublogarithmic approximation for telephone multicast: path out of jungle, *Proc. of Symposium on Discrete Algorithms, SODA'03*, Baltimore,

Maryland, pp. 76–85, 2003.

[34] A. M. Farley, Minimal broadcast networks, *Networks*, 9:313–332, 1979.

[35] A. M. Farley. Minimum-time line broadcast networks, *Networks*, 10:59-70, 1980

[36] A. M. Farley. Broadcast time in communication networks, *SIAM J. Applied Math.*, 39(2):385-390, 1980.

[37] A. M. Farley, S. T. Hedetniemi, S. Mitchell, A. Proskurowski, Minimum broadcast graphs, *Discrete Math.* 25, pp. 189–193, 1979.

[38] U. Feige, D. Peleg, P. Raghavan and E. Upfal, Randomized broadcast in networks, *Proc. of International Symposium on Algorithms, SIGAL'90*, pp. 128–137, 1990.

[39] R. Feldmann, J. Hromkovic, S. Madhavapeddy, B. Monien, and P. Mysliwietz, Optimal algorithms for dissemination of information in generalized communication modes, *Discrete Applied Mathematics*, 53(1-3):55-78, 1994.

[40] G. Fertin and A. Raspaud. A survey on Knodel graphs, *Discrete Applied Mathematics*, 137(2):173-195, 2004.

[41] G. Fertin, A. Raspaud, O. Sykora, H. Schroder, and I. Vrto. Diameter of Knodel graph, *26th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2000)*, Lecture Notes in Computer Science, 19(28):149-160. Springer-Verlag, 2000.

[42] P. Fraigniaud, A note on line broadcast in digraphs under the edge-disjoint paths mode, *Discrete Applied Mathematics*, 144(3):320-323, 2004.

[43] P. Fraigniaud, E. Lazard, Methods and problems of communication in usual networks, *Discrete Appl. Math.* 53:79–133, 1994.

[44] P. Fraigniaud, S. Vial, Approximation algorithms for broadcasting and gossiping, *J. Parallel and Distrib. Comput.* 43(1):47–55, 1997.

[45] P. Fraigniaud, S. Vial, Heuristic Algorithms for Personalized Communication Problems in Point-to-Point Networks, *Proc. of the 4th Colloquium on Structural Information and Communication Complexity, SIROCCO'97*, pp. 240–252, 1997.

[46] P. Fraigniaud, S. Vial, Comparison of Heuristics for One-to-All and All-to-All Communication in Partial Meshes, *Parallel Processing Letters*, 9(1), pp. 9–20, 1999.

[47] L. Gargano, A. L. Liestman, J. G. Peters, D. S. Richards. Reliable broadcasting, *Discrete Applied Mathematics*, 53(1-3):135–148, 1994.

[48] L. Gargano, U. Vaccaro. Minimum time broadcast networks tolerating a logarithmic number of faults, *SIAM J. Disc. Math.*, 5(2):178–198, 1992.

[49] L. Gargano, U. Vaccaro, On the construction of minimal broadcast networks, *Networks* 19, pp. 673–689, 1989.

[50] M. Grigni, D. Peleg, Tight bounds on minimum broadcast networks, *SIAM J. Discr. Math.* 4, pp.207-222, 1991.

[51] H. Haroutunian. Minimal broadcast networks, in *Fourth International Colloquium on Coding Theory*, Dilijan, Armenia, pp. 36–40, 1991.

[52] H. A. Harutyunyan, An efficient vertex addition method for broadcast networks, *Internet Mathematics* 5(3),197–211, 2008.

[53] H. A. Harutyunyan. Minimum Multiple Message Broadcast Graphs, *Networks*, 47(4):218-224, 2006.

[54] H. A. Harutyunyan. Multiple broadcasting in modified Knodel graphs, *7th International Colloquium on Structural Information and Communication Complexity (SIROCCO2000)*, LAquila, Italy, pp. 157-166, 2000.

[55] H. A. Harutyunyan and A. L. Liestman. Messy Broadcasting, *Parallel Processing Letters*, 8(2):149-159, 1998.

[56] H. A. Harutyunyan, A. L. Liestman, More broadcast graphs. *Discr. Appl. Math.* 98:81–102, 1999.

[57] H.A. Harutyunyan,A.L. Liestman. Improved upper and lower bounds for k-broadcasting, *Networks* 37:94-101, 2001.

[58] H. A. Harutyunyan, A. L. Liestman, k-Broadcasting in Trees. *Networks* 38(3), pp. 163–168, 2001.

[59] H. A. Harutyunyan, A. L. Liestman, On the monotonicity of the broadcast function, *Discr. Math.* 262, pp. 149-157, 2003.

[60] H. A. Harutyunyan, A. L. Liestman, K. Makino, and T. C. Shermer, Non-adaptive Broadcasting in Trees, *Networks*, 2009, to appear.

[61] H. A. Harutyunyan, P. Hell, A. L. Liestman. Tight Upper Bounds on Messy Broadcast Time, *Discrete Appliead Mathematics*, to appear, 2010.

[62] H. A. Harutyunyan, A. L. Liestman, B. Shao. A linear algorithm for finding the k-broadcast center of a tree. *Networks*, 53(3): 287–292 (2009)

[63] H. A. Harutyunyan, G. Laza, E. Maraachlian. Broadcasting in necklace graphs, *Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering C3S2E'09*, pp. 253–256, 2009.

[64] H. A. Harutyunyan, E. Maraachlian, Broadcasting in Fully Connected Trees, *International Conference on Parallel and Distributed Systems, ICPADS'09*, 2009.

[65] H. A. Harutyunyan, E. Maraachlian, Linear Algorithm for Broadcasting in Networks with No Intersecting Cycles, *International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA'09*, pp. 296–301, 2009.

[66] H. A. Harutyunyan, E. Maraachlian. Near Optimal Broadcasting in Optimal Triple Loop Graphs, *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications, AINA 2008*, pp 227–233, 2008.

[67] H. A. Harutyunyan, E. Maraachlian, On Broadcasting in Unicyclic Graphs, *Journal of Combinatorial Optimization*, 16:307-322, 2008.

[68] H. A. Harutyunyan, E. Maraachlian. Linear Algorithm for Broadcasting in Unicyclic Graphs, *Proc. 13th Annual COCOON 2007*, LNCS, pp. 372-383, 2007.

[69] H. A. Harutyunyan, E. Maraachlian. Broadcasting in Optimal Bipartite Double Loop Graphs *Proceedings of the conference on Information Visualization, IV 2006*, pp.521–528, 2006.

[70] H. A. Harutyunyan, C. D. Morosan. The spectra of Knodel graphs, *Informatica an International Journal of Computing and Informatics*, 30(3):295-299, 2006.

[71] H. A. Harutyunyan, Calin D. Morosan. On the minimum path problem in Knodel graphs. *Networks* 50(1),pp. 86–91, 2007.

[72] H. A. Harutyunyan, S. Kamali. Optimum Broadcasting in Complete Weighted-Vertex Graphs. *Proceedings of the 36th Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM'10*, pp. 489–502, 2010.

[73] H. A. Harutyunyan, S. Kamali. Efficient Broadcasting in Networks with Weighted Nodes. *Proceedings of the 2008 14th IEEE International Conference on Parallel and*

Distributed Systems ICPADS'08, pp. 879–884, 2008.

[74] H. A. Harutyunyan, S. Kamali. Broadcasting in Weighted-Vertex Graphs, *Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications ISPA'08* pp. 301–307, 2008.

[75] H. A. Harutyunyan, B. Shao, An Efficient Heuristic for Broadcasting in Networks, *Journal of Parallel and Distributed Computing* 66(1), pp. 68–76, 2006.

[76] H. A. Harutyunyan and P. Taslakian. Orderly broadcasting in a 2D torus, *Proceedings of the Eighth International Conference on Information Visualisation (IV04)*, pp. 370-375, 2004.

[77] H. A. Harutyunyan and X. Xu. New Construction of Broadcast Graphs, *11th International Conference Information Visualization*, pp. 751–756, 2007.

[78] S. M Hedetniemi, S.T. Hedetniemi, A.L. Liestman, A survey of gossiping and broadcasting in communication networks, *Networks*, 18:319–349, 1988.

[79] M-C. Heydemann, J. Opatrny, and D. Sotteau. Broadcasting and spanning trees in de Bruijn and Kautz networks, *Discrete Applied Math.*, vol 37 - 38, pp. 297-317, 1992.

[80] J. Hromkovic, C.-D Jeschke, and B. Monien, Optimal algorithms for dissemination of information in some interconnection networks, *Algorithmatica*, 10(1): 24–40, 1993.

[81] J. Hromkovic, R. Klasing, B. Monien, R. Peine, Dissemination of information in interconnection networks, *Combinatorial Network Theory*, D.-Z. Du, D.F.Hsu(eds.), Kluwer Academic Publishers, 1996, pp. 125–212.

[82] J. Hromkovic, R. Klasing, and E.A. Stohr. Dissemination of information in vertex-disjoint paths mode, *Computers and Artificial Intelligence*, 15(4):295-318, 1996.

[83] J. Hromkovic, R. Klasing, W. Unger, and H. Wagener. Optimal algorithms for broadcast and gossip in the edge-disjoint path modes, *Proceedings of the 4th Scandinavian Workshop on Algorithm Theory*, pp.219-230, 1994.

[84] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger. Dissemination of Information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-Tolerance. Springer, 2005.

[85] A. Jakoby, R. Reischuk, C. Schindelhauer, The Complexity of Broadcasting in Planar and Decomposable Graphs, *Discrete Applied Mathematics*, 83:179–206, 1998.

[86] K. Jansen, H. Muller. The Minimum Broadcast Time Problem for Several Processor Networks, *Theoretical Computer Science*, 147(1,2):69–85, 1995.

[87] D. Johnson and M. Garey, Computers and Intractability: A Guide to the Theory of NP-Completeness (Freeman, San Francisco, CA, 1979)

[88] J.-H. Kim and K.-Y. Chwa. Optimal broadcasting with universal lists based on competitive analysis, *Networks*, 45(4):224-231, 2005.

[89] L. Khachatrian and H. Haroutunian, Construction of new classes of minimal broadcast networks, *Proceedings 3rd International Colloquium on Coding Theory, Dilijan, Armenia*, pp. 69–77, 1990.

[90] L. Khachatrian and H. Haroutunian. On optimal broadcast graphs, *Fourth International Colloquium on Coding Theory*, Dilijan, Armenia, pp. 65-72, 1991.

[91] R. Klasing, B. Monien, R. Peine, E. A. Stöhr, Broadcasting in butterfly and deBruijn networks, *Discrete Applied Mathematics* 53:183–197, 1994.

[92] S. Khuller, Y. A. Kim. Broadcasting in Heterogeneous Networks, *Algorithmica*, 48(1):1–21, 2007.

[93] S. Khuller, Y. A. Kim. On broadcasting in heterogenous networks. *Proc. of Symposium on Discrete Algorithms, SODA'04*, pp. 1011–1020, 2004.

[94] S. Khuller, Y. A. Kim, Y-C. Wan. On generalized gossiping and broadcasting. *J. Algorithms* 59(2): 81–106, 2006.

[95] S. Khuller, Y. A. Kim, Y-C. Wan. Broadcasting on networks of workstations. *Symposium on Parallel Algorithms SPAA'05*, pp. 279–288, 2005.

[96] S. Khuller, Y. A. Kim, G. J. Woeginger. Approximation Schemes for Broadcasting in Heterogenous Networks, *APPROX-RANDOM 2004*, pp. 163–170, 2004.

[97] S. Khuller, Y. A. Kim, Y-C. Wan. On Generalized Gossiping and Broadcasting (Extended Abstract). *European Symposium on Algorithms, ESA'03*, pp. 373–384, 2003.

[98] G. Kortsarz and D. Peleg, Approximation algorithms for minimum time broadcast, *SIAM J. Discrete Math.* 8:401–427, 1995.

[99] W. Knodel. New gossips and telephones, *Discrete Mathematics*, 13, p. 95, 1975.

[100] R. Labahn. A minimum broadcast graph on 63 vertices, *Discrete Appl. Math.*, 53:247-250, 1994.

[101] R. Labahn, Extremal broadcasting problems, *Discrete Applied Mathematics*, 23:139–155, 1989.

[102] T. Leighton, Introduction to Parallel Algorithms and Architectures: Array-Trees-Hypercubes, Morgan-Kaufmann Publishers, San Mateo, California, 1992.

[103] C. Li, T. E. Hart, K. J. Henry, and I. A. Neufeld. Average-case messy broadcasting, *Journal of Interconnection Networks*, 9(4),pp 487–505, 2008.

[104] A. Liestman and J.Peters, Broadcast networks of bounded degree, *SIAM, J. Disc. Math*, 1:531–540, 1988.

[105] A. Liestman, T. Shermer, and M. Suderman. Broadcasting multiple messages in hypercubes, *International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 00)*, pp 274-281, 2000.

[106] M. Maheo and J.-F. Sacle. Some minimum broadcast graphs, *Discrete Appl. Math.*, 53:275-285, 1994.

[107] M. Middendorf, Minimum broadcast time is NP-complete for 3-regular planar graphs and deadline 2, *Information Processing Letters*, 46(6): 281–287, 1993.

[108] A. Pelc. Fault-tolerant broadcasting and gossiping in communication networks, *Networks*, 28(3):143–156, 1996.

[109] David Peters, Bounds of Communication Problems in Interconnection Networks under a Linear Cost Model, *PhD Thesis*, Simon Fraser University, Canada, 1995.

[110] R. Ravi, Rapid Rumor Ramification. Approximating the minimum broadcast time, *Proc. of 35th Symposium on Foundation of Computer Science, FOCS'94*, pp. 202–213, 1994.

[111] J.-F. Sacle. Lower bounds for the size in four families of minimum broadcast graphs, *Discrete Math.*, 150:359-369, 1996.

[112] P. J. Slater, E. J. Cockayne, S. T. Hedetniemi. Information dissemination in trees, *SIAM J.Comput.* 10(4), pp. 692–701, 1981.

[113] P. Scheuerman, G. Wu. Heuristic Algorithms for Broadcasting in Point-to-Point Computer Network, *IEEE Transactions on Computers* C-33(9), pp. 804–811, 1984.

[114] C. Schindelhauer. On the Inapproximability of Broadcasting Time, Proc. of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'00), pp.226–237, 2000.

[115] J. L. Traff and A. Ripke, Optimal broadcast for fully connected processor-node networks. *J. Parallel Distrib. Comput.*, 68(7),pp 887–901, 2008.

[116] J. A. Ventura and X. Weng. A new method for constructing minimal broadcast networks, *Networks*, 23:481-497, 1993.

[117] M. X. Weng and J. A. Ventura. A doubling procedure for constructing minimal broadcast networks, *Telecomm. Syst.*, 3:259-293, 1995.

[118] J. Xiao and X. Wang. A research on minimum broadcast graphs, *Chinese J. Computers*, 11:99-105, 1988.

[119] E. W. Zegura, K. Calvert, S. Bhattacharjee, How to model an internetwork, *Proc. The IEEE Conf. on Computer Communications, INFOCOM'96*, San Francisco, CA, 1996.

[120] J.-G. Zhou and K.-M. Zhang. A minimum broadcast graph on 26 vertices, *Appl. Math. Lett.*, 14:1023-1026, 2001.