BROADCASTING IN WEIGHTED-VERTEX GRAPHS

SHAHIN KAMALI

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

AUGUST 2008

Your file Votre référence
ISBN: 978-0-494-45337-7
Our file Notre référence
ISBN: 978-0-494-45337-7

Abstract

Broadcasting in Weighted-Vertex Graphs

Shahin Kamali

In this thesis, we present a new model for information dissemination in communication networks. The model is defined on networks in which nodes are assigned some weights representing the internal delay they should pass before sending data to their neighbors. The new model, called weighted-vertex model, comes to have real world applications in parallel computation and satellite terrestrial networks. Broadcasting in weighted-vertex model is a generalized version of classical broadcasting problem, which is NP_Complete. The problem remains NP_Complete in some classes of weighed-vertex graphs. We show existence of approximation algorithms for the broadcasting problem in weighted vertex model, as well as better approximations for specific subclasses of weighted graphs. In addition we study broadcasting in complete weighted graphs and present an algorithm for finding the optimum solution in this case. Broadcasting in complete graphs with uniform weights is studied separately. Finally we introduce some heuristics for the problem and compare their performance using computer simulations.

Acknowledgments

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

I owe everything to my mom, my dad, and my brother Shahab. Without their endless love and support I wouldn't have been here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, telecommunications networks have become essential in many aspects of modern society. Specifically computer networks have dramatically changed our life style. We are deeply influenced by the Internet and different types of local area networks through which data can be sent and received in fraction of a minute.

Computer networks are also required in applications such as scientific simulations and 3D animation which demand an enormous amount of computation. Computers with few processors are not powerful enough to process the huge amount of data at a practical speed. This is the reason that *parallel machines* [70], also known as MPC (Massively Parallel Computer) [72], are built to connect thousands of processors each having its own local memory.

The main purpose of any network, no matter a MPC network or the Internet, is to share and spread information. Communication efficiency becomes particularly important when a computer network supports a distributed file or database system, where a large amount of information needs to be disseminated among the computers in the network. For example in parallel computers, the problem is usually divided into small tasks distributed to a set of nodes which compute concurrently. Since the nodes in a parallel machin do not share

physical memory, they must communicate by passing messages through the network. Consequently, efficient routing of messages is critical to the performance of parallel computers [72].

In general, it is the performance of the information dissemination that often determines the efficiency of a whole network or a parallel system.

One approach to improve the performance of information dissemination is to create network topologies in which information can be transmitted quickly. *Complete graphs* and *Hypercube* graphs are two popular communication networks for fast dissemination of information. A complete graph is a graph in which there is a link between any pair of nodes (Figure 1(a)). A hypercube of dimension d is composed of 2^d vertices representing binary strings of length d. Two vertices are connected if their binary representation differs exactly in one bit (Figure 1(b)). Note that the number of links in complete and hypercube graphs is rather high, making them costly for network design. There is an extensive research for creating graphs with the lowest possible number of edges in which the information dissemination can be performed in minimal time (see for example [25, 14, 76, 31, 66, 50, 45, 47, 12, 46]).

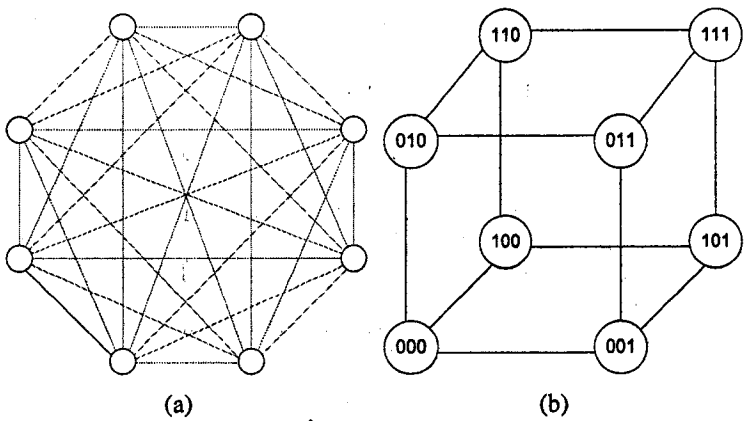


Figure 1: Complete graph (a) an hypercube graph (b) with 8 vertices.

A second approach to enhance the performance of information dissemination is to minimize the delay of information spreading. Providing efficient algorithms for effective data dissemination can be traced back to the following problem:

Problem 1. *There are n ladies, each knowing an item of scandal that is not known to any one else. They communicate by telephone, and whenever two of them make a call, they pass on to each other, as much scandal as they know at the time. How many calls are needed before all ladies know all the scandal? [56]*

This problem is the origin of the *Gossip Problem*, which is also the source of dozens of papers on the information dissemination problems in networks. Gossiping along with its variants constitutes the communication primitives which can be classified as follows:

- *Unicasting* or *one-to-one* communication.
- *Broadcasting* or *one-to-all* communication.
- *Multicasting* or *one-to-many* communication.
- *Gossiping* or *all-to-all* communication.

Broadcasting is a process in which a single message is sent from one member of a network, the originator, to all other members; while in gossiping every member in a network has a message to send to all other members. Multicasting is a generalized version of broadcasting in which originator informs a subset of members of the network.

A *call* is an action in which messages exchange among a vertex and at least one of its neighbors. Information is disseminated in the network via a series of calls over the communication links of the network. Note that the communication is assumed to be synchronous, i.e. it occurs in discrete pulses, called *rounds*. In this way a round is the set of parallel calls in the same time unit.

Normally, a network can be modeled as a graph $G = (V, E)$, where V is the set of vertices (nodes) and E is the set of edges. Two vertices $u, v \in V$ are *neighbors* if there is an edge $e \in E$, such that $e = (u, v)$. The degree of a vertex v, denoted by $deg(v)$, is the number of neighbors of v. The degree of a graph G is the maximum degree among all vertices of the graph. The distance between a vertex v and a vertex w, denoted by $dist(v, w)$, is the length of the shortest path between v and w. For message dissemination purposes, it is natural to assume that the network is represented by a connected graph.

Given a graph, an algorithm of a communication problem generates a communication *scheme*, which is a sequence of communication rounds. A scheme can be presented as $< E_1, E_2, \cdots, E_r >$ in which E_i is a set of calls in the form of $v_\alpha \rightarrow v_\beta$ at time unit i.

The number of communication rounds is usually considered as the criteria for measuring the efficiency of a scheme. In the case of broadcasting, the number of rounds is called the *broadcast time* of the scheme. Given a graph G and originator $u \in V(G)$, an optimum broadcast scheme for originator u is a scheme with minimum broadcast time. Also the broadcast time of u is defined to be the broadcast time of an optimum scheme of u.

In the literature, sometimes terms like *broadcast schedule* or *broadcast algorithm* are used instead of broadcast scheme. We use these terms in some parts of this thesis as well. Also, the rounds are usually considered as time intervals. In this way when we say 'at time t' we mean in round number t.

1.1 A Review on the Models of Information Dissemination

Once the communication problem is defined, we need to describe the communication model which precisely defines what may happen in a round.

Regarding the communication link level, two types of models can be considered:

- *One-way mode* or *half-duplex*

- *Two-way mode* or *full-duplex*

Given two neighbor vertices v and w in a graph, under one-way mode, only one message can travel between v and w, either from v to w or from w to v; while under two-way mode two messages can use the link at the same time [60]. In this thesis we are mainly concerned with the communication models defined under two-way mode.

Dissemination models can also be classified based on the number of links employed by an informed node in one round:

- *1-port* or *whispering pattern* in which a node can only use one of its links in one round.
- *n-port* or *shouting pattern* in which a node can use all of its links in parallel and in one unit of time.
- *k-port* also referred as *k-broadcasting*, which is an intermediate pattern in which a node v can use $k < deg(v)$ of its links in parallel and in one time unit.

Among these patterns, 1-port model seems to have more applications. The k-port pattern is also intensively studied in [69, 67, 48, 49, 51, 54, 77, 39]. *Radio Model* is a sample of n-port model in which a node can either send a message to all of its neighbors or not send it at all, but it can not transmit to a strict subset of neighbors. Furthermore, a node can receive the message from precisely one neighbor in a certain round, otherwise the message gets corrupted [29]. In most of the models we present in the rest of this section, the communication mode is assumed to be 1-port.

Another issue in characterizing communication models is the time required for a message to be prepared, to travel along an edge, and to be received. There are two widely used models addressing this issue:

- The constant model, in which the time needed to transmit and receive a message is a constant value T.

- The linear model, in which the time needed to communicate is modeled as $T = \beta + L\tau$, where β is the cost of preparing the message, L is the length of the message, and τ is the propagation time of a data unit length.

Most papers on broadcasting adopt the constant model, since the linear model can be reduced to the constant model by splitting large messages to the shortest possible messages. Nevertheless, there are some specific results regarding the linear model for broadcasting (see [10]).

Several combinations and variations of these patterns have been considered. These include simultaneous sending to all neighbors but sequential receiving, or the concurrent sending to one neighbor and receiving from one (possibly different) neighbor. In the rest of this section, some of the developed models are reviewed. In providing some parts of this review, we have referred to [71].

- **Telephone Model (Classical Model)**:

  Telephone model is usually considered as the classical Model of information dissemination. When we do not precisely mention the communication model, the model is assumed to be the telephone model.

  The scandal problem introduced above adheres to the telephone model. Since in that problem ladies communicate via telephone, the term 'telephone' is adopted to the model; However this model can be applied for any network. In this thesis, we preferably use the term 'classical' to distinguish this model from the other models. Classical model is based on the following assumptions:

  - 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.

From these assumptions, it is clear that the classical model is an instance of two-way model in which the communication pattern is whispering. Also it is a constant model in which the message transmit time is $T = 1$.

The classical model of broadcasting has been intensively discussed. We review some of the results in the next section. An important result is about the hardness of problem, as it is proved that the broadcasting problem is NP_Complete under this model [42]. The classical model has been the inspiration root of most of the models developed afterward.

- **Telegraph model**

Telegraph Model is an abstract version of one-way mode model. In this way, in each round, information can be transferred from node v to one of its neighbor, say w, meanwhile w can not send its information to v at the same time.

Except being a one-way mode communication model, the other assumptions behind telegraph model is the same as the classical model. However this small difference makes telegraph model much more tricky. To have an idea of difficulty, consider complete and hypercube graphs. While under the classical model the complexity of gossiping for these graphs is easy to check, under telegraph model, the gossiping time is not known for complete graph K_n for all values of n and the complexity of gossiping in hypercubes is totally unknown [40].

- **Line model**

Line model assumes long distance calls between non neighboring processors. In this sense, in every round of communication, the information is transmitted from the informed nodes to uninformed nodes through paths that may have length greater than one. If these paths be composed of disjoint vertices, the model will be *vertex disjoint*

path mode broadcasting. There are two flavors of this model considering either that an end-node broadcasts its data to all other nodes along the path, or one of the end-nodes sends data to the other end-node and the nodes in between do not read the message sent [32, 34, 68, 61, 16, 35, 36, 38].

In line model, if the communication paths be edge disjoint (and not necessarily vertex-disjoint), the result would be edge disjoint path mode broadcasting which is also investigated in several papers [32, 34, 62, 37, 35]. This model is quite similar to the wormhole routing model employed for the analysis of permutation routing [22, 65, 2].

A central assumption to all models described above is that both sender and receiver are busy during the whole sending process; that is only after the receiver received the message, both ends may send the message to other nodes. More realistic models are Postal model [7] and LogP model [21]. The idea there is that the sender may send another message before the current message is completely received by the receiver, and the receiver is free during the early stages of sending process. In this way the underlying communication network can be modeled by a weighted graph. However in these models the exact structure of the network is totally ignored and it is assumed that the network creates a complete communication graph between nodes (processors).

- **Postal model**

  Postal model incorporates a latency parameter $\lambda \geq 1$ that measures the inverse of the ratio between the time that it takes for a node to send a message and the time that passes until the recipient of the message receives it. In this way we can consider postal model as a model of broadcasting in complete graphs such that a uniform delay λ is associated to each edge (For $\lambda = 1$ the postal model reduces to the classical model in fully-connected networks).

In [7], an algorithm is presented for broadcasting one message in systems with n processors and communication latency λ, whose running time is $\Theta(\frac{\lambda \log(n)}{\log(\lambda+1)})$. This algorithm is proved to create the optimum solution for the problem in Postal model.

- **Generalized postal model**

  In the basic postal model, it is assumed that the communication delay (λ) is constant. In [8], the postal model is enhanced by removing this assumption and different links may have different weights. Consequently, some edges can be *disabled* by assigning large weights to them. In this way we can define some kind of *topology* for the graphs; which means that the model applies for the graphs that are not necessarily fully connected. Hence, the classical model reduces to generalized postal model. As a result, broadcasting under this model is NP_Complete. In [8], heuristic algorithms are presented for broadcasting under this model.

- **LogP model**

  LogP Model is another generalization of postal model. This model is mainly applied for the parallel computers having a point-to-point message transmission link between any two processors. In this way the network topology is considered to be a complete graph. In LogP a parallel machine is described by four parameters:

  - L, the *latency*, or maximum delay, associated with delivering a message.
  - o, the *overhead*, representing the length of time for which a processor is busy during the transmission or reception of a message.
  - g, the *gap*, a lower bound on the time between the transmission of successive messages, or the reception of successive messages at the same processor.
  - P, the number of processor-memory pairs.

  The parameters L, o, g are measured in units of processor cycles (rounds). Postal model turns out to be an instance of LogP model with $g = 1$ and $o = 0$. LogP model

also specifies that the network has a *finite capacity* such that at most $\lceil L/g \rceil$ messages can be in transit from any processor to any processor at any time. In general, LogP model can be considered as a 'detailed' model which tries to reflect the critical technology trends underlying parallel computers. As a result, presenting and analysis of algorithms under this model is rather hard.

- **Heterogeneous postal model**

  In [4] heterogeneous postal model is defined as another generalization of postal model. Consider node v sends the message to node w at time 0 and w receives the message at time λ_{vw}. In this model, the assumption is that v is free to send a new message at time s_v, and w is free from time 0 to $\lambda_{vw} - s_v$. The value λ_{vw} is called the delay of link (v, w), and s_v is the switching time of v. When the delay and switching times are both equal to 1, we obtain postal model. While the broadcasting problem is NP_Complete under this model, in [4] an $O(logk)$ approximation algorithm is presented for the problem.

- **Multiport model**

  This model is a sample of k-broadcasting in which in one round, each node can send k messages to k processors and receive k messages from k processors. In [6], an algorithm is presented for broadcasting in multiport model that produces schemes which are optimum with an additive term of one for any number of processors, any number of messages, and any value for k.

- ***(i,j)* mode broadcasting**

  In the models obeying this mode, in each round, a node can send a message to i neighbors and receive messages from j neighbors. Therefore, the two-way mode can be regarded as a $(1, 1)$ mode with the additional constraint that the edges for receiving and transmitting are the same. The *(i,j)* mode broadcasting has been investigated in

[30].

- **Messy broadcasting**

  The messy broadcasting model has been introduced in [1]. Unlike the previously presented models, messy broadcasting is concerned with analyzing the worst case performance of the broadcast schemes. In other words, the messy broadcasting model is looking for upper bounds in the broadcast time, following the constraints below:

  - One node knows only its neighbors;
  - The originator or the time slot is not known;
  - There is no coordinating leader;
  - 1-port, constant model is assumed.

  This model is suitable for networks in which the memory of nodes is insufficient to keep a coordinated protocol. The model is studied in [19, 43, 55, 44].

1.2 Results for Broadcasting in Classical Model

In this section we review some of the results related to the classical model of information dissemination. Some of these results can be extended to the generalized model introduced in Chapter 2.

In the rest of this thesis we mainly focus on broadcasting instead of multicasting and gossiping. The reason is that most of the results on broadcasting can be easily generalized to multicasting, while studying broadcasting makes notifications easier. Also the following theorem shows that it is not needed to study gossip problem separately.

Theorem 1. *In any communication model obeying two-way communication mode, the gossip problem can be reduced to broadcast problem [60].*

Proof. Consider the *information accumulation problem.* This problem is the same as the broadcast problem in which instead of informing all the vertices, a node v needs to receive separate pieces of information from all the nodes. In this way, if $< E_1, E_2, \cdots, E_r >$ be a broadcast scheme for a node in a two-way model, then $< E_r, E_{r-1}, \cdots, E_2, E_1 >$ would be an accumulation scheme for the same node with the same time.

In the gossiping problem, all the nodes need to accumulate the message. As a result, a gossiping scheme can be represented as a set of n accumulation schemes performing in parallel. We conclude that the gossiping time, in any communication model, is equal to accumulation time. Also as mentioned before, in two-way models, broadcasting and gossiping times are equal. We conclude that in any two-way model (including the classical model) the gossiping problem reduces to broadcasting problem. □

Note that broadcast problem can be defined for both directed and undirected graphs. However in most applications the network is modeled by an undirected network. In this thesis, by notation 'graph' we always mean undirected graph.

It is easy to verify that under the classical model, two trivial lower bounds hold for the minimum broadcast time. The first one is $\lceil \log n \rceil$, where n denotes the number of nodes in the graph, and the second one is the maximum distance from the source to any of the other nodes. The second bound is rather obvious, while the first bound holds because the number of informed nodes can be at most doubled in each round.

As mentioned earlier, given a graph $G = (V, E)$ and an originator $u \in V$, it is NP_Hard to find the optimum broadcast scheme. The proof involves a reduction from 3D_Matching problem to the broadcast problem [42]. The following is an important byproduct of this result:

Theorem 2. *The broadcast problem is NP_Complete in directed graphs.*

Proof. Broadcasting in undirected graphs can be easily reduced to broadcasting in directed graphs. We just need to replace each undirected edge (v, w) with two directed edges, one

from v to w and one from w to v. □

In addition it is proved that the problem remains NP_Hard for some restricted types of graphs including bounded degree graphs [13], chordal graphs [64], planar bipartite graphs [64], planar and decomposable graphs [63].

Finding good approximations has been intensively studied for broadcasting in undirected graphs [68, 75, 4, 41]. The best theoretical upper bound is presented in [28] involving algorithms approximating the broadcast time with a factor of $\log(n)/\log\log(n)$ in which n is the number of vertices.

k-multicast problem has also been considered in the literature. Having a multicast set of size k, the best approximation is the one presented in [28] with an approximation ratio of $\log(k)/\log\log(k)$. This algorithm is actually the same as the one presented for broadcasting which is generalized to multicasting; if we set the size of multicast set to be the same as the number of vertices of graph ($k = n$), we come to the same ratio $\log(n)/\log\log(n)$ for the broadcast problem.

In [26] it has been shown that it is NP_Hard to find a broadcast scheme within an approximation ratio of $3 - \epsilon$ for any $\epsilon > 0$.

The problem appears to be much harder in directed graphs; most of the previous approximations can not be extended to directed graphs. However the algorithm presented in [26] applies to directed graphs and gives a logarithmic approximation to the problem. Multicasting in directed graphs has also been studied in [28, 27]. The algorithm presented in [27] constructs a schedule with $O(b^* \log k + k^{1/2})$ in which k is the size of multicast set and b^* is the optimum broadcast time.

The broadcast process has also been studied for other particular network topologies, such as trees [79], unicyclic graphs [52, 53], grids [33, 78, 18, 81], Kautz graphs [58], pancake and star graphs [15], chordal rings [20], k-trees [23], planar graphs [57, 63, 64] recursive circulants [73], banyan-hypercube [11], de Bruijn, shuffle-exchange and similar

networks [74].

1.3 Thesis Contribution

Although there has been a huge amount of research on classical model of broadcasting, this model is not always appropriate for all real world problems. In this thesis we present a new communication model called weighted-vertex model.

In Chapter 2, the weighted-vertex model is defined and some of its basic properties are discussed. In this model, the network is modeled by a graph in which the vertices are assumed to be weighted. It is shown that weighted-vertex model can be applied on some real world problems while remaining interesting in theoretical aspects. We generalize some of the properties of the classical model to the weighted-vertex model. We also introduce uniform graphs, binary graphs, heavy graphs, even graphs and odd graphs as specific types of weighted graphs in which the broadcasting problem seems to have a more interesting nature. Finally we present an efficient algorithm for broadcasting in weighted-vertex trees.

As a generalized version of classical model, the broadcasting problem is NP_Complete in weighted-vertex model. In Chapter 3, we present an approximation algorithm for the problem. To do so, we reduce broadcasting in weighted-vertex graphs to multicasting in directed unweighted graphs. In addition we present better approximations for broadcasting in even graphs, binary graphs and heavy graphs.

Chapter 4 is dedicated to the study of the broadcasting problem in fully connected weighted graphs. We present a polynomial algorithm for producing optimum broadcast schemes in complete weighted graphs. We also study the problem in complete graphs with uniform weights, which gives a lower bound for the broadcast time in weighted graphs in general case.

In Chapter 5, we present three heuristics for the problem. We employ a modified version of Dijkstra's algorithm, a greedy algorithm, and also an evolutionary algorithm. Using

computer simulations we compare the performance of these three algorithms in different weighted-vertex graphs.

Chapter 2

Weighted-Vertex Model

2.1 Model Definition

As mentioned earlier, the classical model of broadcasting does not always fit the requirements of real-world situations. This is the reason that several other models with different characteristics are presented.

Here we enhance the classical model to the graphs with weighted vertices, resulting a new model called weighted-vertex model. The weights assigned to the vertices may have different meanings. For example each node may represent a smaller network such that the assigned weight represents an upper bound for the broadcast time of that network. In parallel machines, the weights assigned to processors can be interpreted as a kind of internal process each processor needs to perform before starting informing its neighbors (this is the reason that sometimes we regard the weights as *internal process* or *internal delay*). The weights can also be considered as the delay for receiving the message due to input device limitations.

Another application of the weighted-vertex model is in Satellite-Terrestrial (ST) networks. These are networks in which a large number of nodes are interconnected by both

satellite and terrestrial networks [3]. ST networks can effectively support multicast services; however the performance of these networks is not high when they are modeled by regular unweighted graphs. The reason is that the cost of satellite nodes is more than terrestrial nodes. To resolve this problem, ST networks has been modeled by weighted-vertex graphs in which the vertices representing satellite nodes have positive integer weights, while the vertices representing terrestrial nodes have 0 as their costs. The weight assigned to the satellite nodes is the sum of the costs of up-links and down-links connecting them to the terrestrial network. By including these vertices in the broadcast tree, their costs are included in the costs of trees [3, 82].

While defining a new model, it may come to mind to put the weights on both edges and vertices. However putting weights on the edges is not interesting in the sense that any edge of weight w can be replaced with a path of length w to get the same problem. In the rest of this thesis, whenever we use term weighted graphs, we mean graphs with weights on the vertices (and not on the edges). As mentioned above, sometime we refer these weights as *internal process* or *internal delay* of the vertices.

Definition 1. *Let $G = (V, E)$ be an undirected graph where V is a set of n nodes and E is the set of communication links and let $u \in V$ be the originator node. We associate with each node $v \in V$ a non-negative integer weight $w(v)$. These weights represent the delay of a node to perform some process after receiving the message. Suppose that node v* 'receives' *the message at time t from one of its neighbors. After receiving the message, v should wait for $w(v)$ rounds and then start informing its uninformed neighbors. This is the time in which v* 'completes' *handling its delay. In this way, the first neighbor of v receives the message at time unit $t(v) + w(v) + 1$.*

It is not hard to verify that weighted-vertex model is a novel model which differs from all models reviewed in Chapter 1. Note that the weights assigned to the nodes are always considered to be non-negative integers. When the weights of all vertices are 0, the model

is equivalent to the classical model.

The weighted-vertex model applies for digraphs as well; however in this thesis we always consider that the weighted graphs are undirected. Multicasting and gossiping could also be considered under the new model.

In weighted graphs, similar to unweighted graphs, any broadcast scheme can be represented by a spanning tree which is rooted at the originator. Later we show the existence of an effective algorithm for broadcasting in trees under weighted-vertex model. This algorithm enables us to represent broadcast schemes with spanning trees.

Note that in the broadcast tree of an instance (u, G) of broadcast problem, the root (originator) and the leaves have some weights that are considered as a part of the broadcast time. So when a leaf v receives the message, the broadcasting process is not complete before an additional $w(v)$ time units. Figure 2 shows a broadcast tree while node A is the originator. In this scheme A receives the message at time $t = 0$. It is busy during next 5 rounds until it completes its delay at time $t = 5$. Then A informs B at time $t = 6$, C at time $t = 7$, D at time $t = 8$ and E at time $t = 9$. The same way B is busy in time period $[6, 8]$. Afterward, F receives the message through B at time $t = 9$ and completes after 3 rounds delay when it sends the message to K. The broadcasting completes when K completes its internal delay at time $t = 18$.

We can also present a broadcast scheme in weighted-vertex model as a sequence of rounds $< E_1, E_2, \cdots, E_r >$ in which E_i is a set of 'calls' and 'internal delay units' in the form of $v_\alpha \rightarrow v_\beta$ and $v_\alpha \downarrow$ respectively. In this way $v_\alpha \downarrow \in E_i$ means that node v_α passes one unit of its internal delay in round i. We also denote the final internal delay unit of a vertex v_α by $v_\alpha \downarrow_*$.

Definition 2. *A broadcast (multicast) scheme is a* non-lazy *scheme iff any vertex v after receiving the message and completing its internal delay, in all rounds before the broadcast process completes, informs one of its uninformed neighbors.*

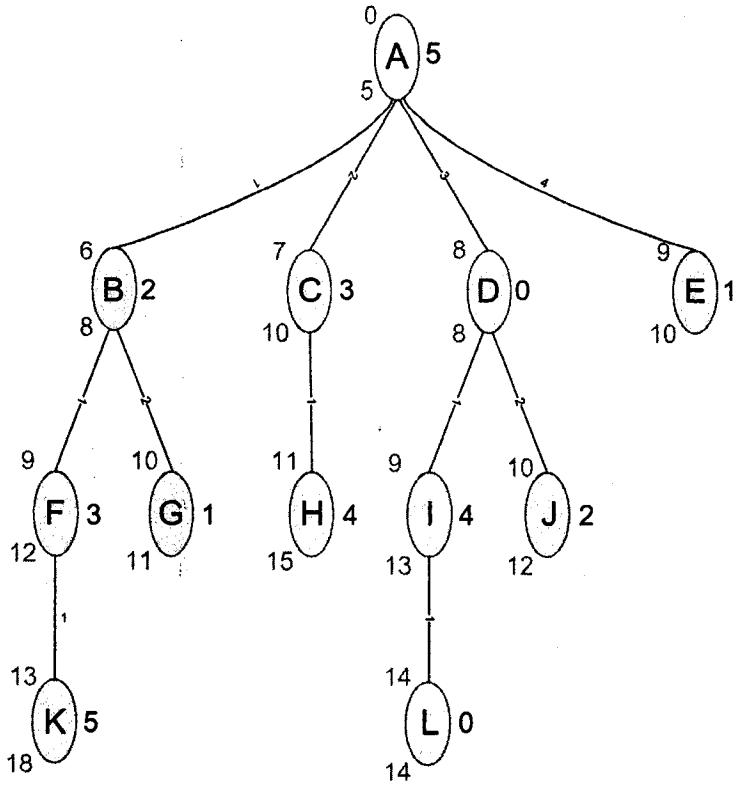


Figure 2: A broadcast tree rooted at originator A. The numbers in the right side of nodes indicate their weights; while the numbers in left indicate the times in which nodes receive the message (top) and complete their delay (bottom).

It is easy to see that for any broadcast scheme, there is an equivalent non-lazy scheme with lower or equal broadcast time. From now, whenever we talk about a scheme, we mean a non-lazy scheme.

Note that, in many concepts the weighted-vertex model is similar to the classical model of broadcasting:

Fact 1. *In the weighted-vertex model of broadcasting:*

- *The communication mode is two-way*
- *Message transition pattern is 1-port (whispering)*

- *The model can be considered as a sample of constant model in the sense that the time needed to transmit a message through a link is the constant 1; although the time needed to handle the message varies for different nodes.*

The classical model is also a two-way, 1-port and constant model. This is the reason that sometimes instead of using the term 'weighted-vertex model of broadcasting' we use 'broadcasting in graphs with weighted vertices'.

Theorem 3. *In the weighted-vertex model, the gossipping problem reduces to broadcasting problem.*

Proof. Since the model is a two-way model, we can apply Theorem 1 and conclude that in weighted-vertex model gossiping can be reduced to broadcasting. □

Here we present some easy lower and upper bounds for the broadcast time in weighted-vertex model. Let $b(u, G)$ denotes the minimum time needed to complete broadcasting from originator $u \in V(G)$. Also let W denote the sum of the weights of all vertices, i.e. $W = \sum_{v \in V} w(v)$.

The following theorem gives an obvious lower bound for $b(u, G)$:

Theorem 4. $b(u, G) \geq d + (d+1)\, w_{min}$ *in which* d *is the maximum distance of* u *to other nodes and* w_{min} *is a lower bound for the weights of nodes.*

In Chapter 4 another lower bound for the broadcast time will be presented.

Definition 3. *Given a weighted graph G, the underlying graph of G, denoted by* G_0*, is a copy of G in which all the weights are removed. More formally,* G_0 *is an unweighted graph with the same vertex and edge sets as G.*

We can present an upper bound for broadcast time in a weighted graph in terms of total weight and broadcast time of the underlying graph:

Theorem 5. *$b(u, G) \leq W + b(u, G_0)$ in which W is the total weight of G, and G_0 is the underlying graph of G.*

Proof. For any node x in G, consider the path P from originator u to x in an optimum broadcast tree for G. So we have $b_G(u, x) = \sum_{v_i \in P} w(v_i) + l(v_i)$ in which $w(v_i)$ is the weight of v_i and $l(v_i)$ is the label of incoming edge to v_i when called by its parent. Note that $b_{G_0}(u, x) = \sum_{v_i \in P} l(v_i)$. So we have $b_G(u, x) = b_{G_0}(u, x) + \sum_{v_i \in P} w(v_i) \leq b_{G_0}(u, x) + W$. Hence, $b(u, G) \leq b(u, G_0) + W$. □

Applying bounds $b(u, G_0) \leq n-1$ and $W \leq n\, w_{max}$, we come to the following results:

Corollary 1. *$b(u, G) \leq W + n - 1$ in which W is the total weight of graph and n is the number of vertices.*

Corollary 2. *$b(u, G) \leq n\, w_{max} + n - 1$ in which n is the number of vertices and w_{max} is an upper bound for the weights of vertices.*

At the end of this section, we present an important result on the hardness of broadcast problem in weighted-vertex graphs:

Theorem 6. *The problem of broadcasting in weighted-vertex model is NP_Complete.*

Proof. As mentioned before broadcasting in classical form is an instance of weighted-vertex broadcasting in which all the vertices of the graph have uniform weight 0. We conclude that the problem is NP_Hard. Also given a scheme, it can be checked in polynomial time if the scheme completes before a specific time limit (depending on how to represent the scheme the broadcast time of the scheme is also given). So the problem belongs to the class NP as well. □

2.2 Subclasses of Weighted Graphs

In this section we present some important subclasses of weighted-vertex graphs. Most of these graphs would be referred in the next chapters:

- **Uniform Graphs**

  Uniform graphs are weighted graphs in which all the vertices have equivalent weights. A uniform weighted graph can be represented by triple (V, E, w_u) in which V and E are the vertex and edge sets respectively and w_u is the integer representing the uniform weight of all vertices. Uniform graphs can be applied for modeling the networks in which all the nodes are homogeneous; which is usually the case in multi-processor systems. The following bounds hold for broadcast time in uniform graphs:

  Theorem 7. *For any originator u in uniform graph G, we have $d + (d+1)W/n \leq b(u,G) \leq b(u,G_0) + W$, in which d is the maximum distance of u to the other vertices, n is the number of vertices, W is the total weight of graph, and G_0 is the underlying network.*

- **Binary Graphs**

  Binary graphs are the weighted graphs in which the weight of all nodes is either 0 or 1. These graphs are very important in theoretical aspects. Note that, in binary graphs, the total weight W is equal to the number of nodes with weight 1. In Chapter 2, we show that broadcasting in general weighted-graphs can be reduced to broadcasting in binary graphs.

- **Heavy Graphs and Light Graphs**

  A heavy graph G is a weighted graph in which for any node $v \in V(G)$ we have $deg(v) \leq w(v)$. In this way the weights of vertices are relatively high. These graphs may have application for modeling the networks in which the underlying communication network is a low-degree graph. For example planar weighted graphs or

weighted hypercubes have a high chance of being heavy graphs.

In the same way, we can define light graphs as the weighted graph in which for any node $v \in V(G)$ we have $deg(v) \geq w(v)$. These graphs may have applications when the communication network is modeled by a dense graph.

By definition, in any heavy graphs $W \geq 2|E|$ in which W is the total weight and E is the number of edges. The same way, in light graphs, $W \leq 2|E|$.

- **Even Graphs and Odd Graphs**

  An even graph is a weighted graph in which the weight of all nodes is even. The same way an odd graph is a graph with odd integers as the weights of vertices. Note that a uniform graph is either an even graph or an odd graph.

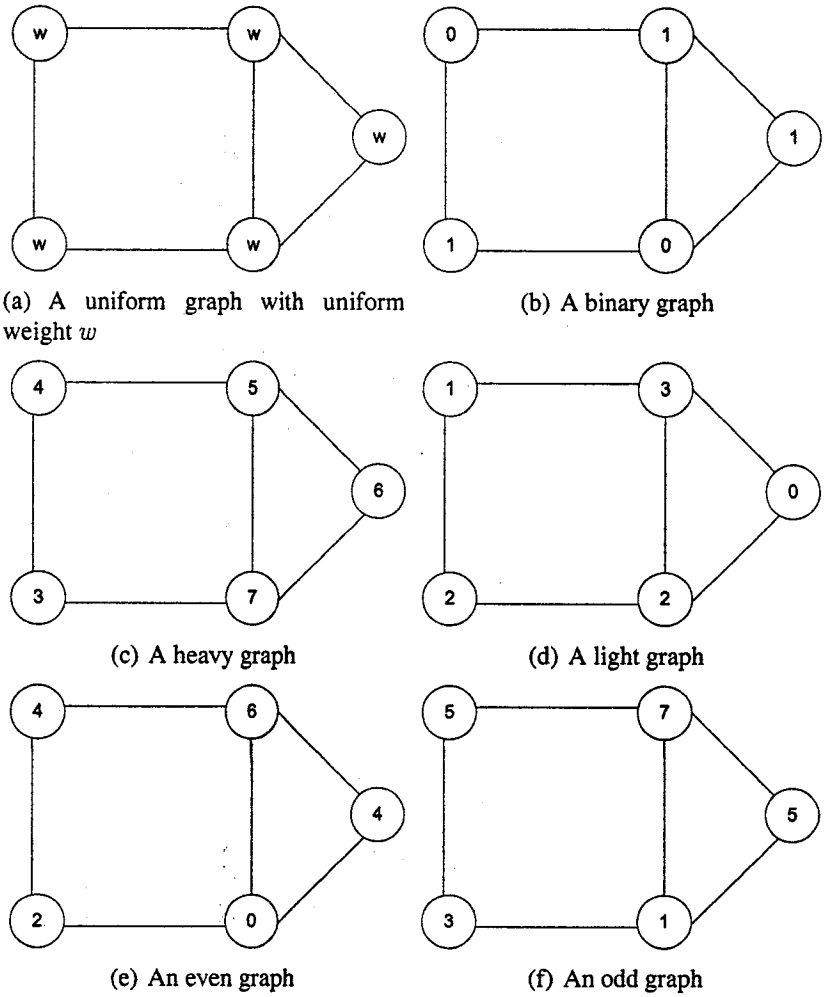
Some samples of the introduced graphs are depicted in Figure 3.

Note that unweighted graphs are specific instances of the uniform graphs, binary graphs, light graphs, and even graphs. Consequently, the broadcast problem is NP_Complete in these four subclasses of weighted graphs. However at this stage, we don't have any NP_Completeness result for broadcasting in heavy graphs and odd graphs.

2.3 Weighted Broadcasting in Trees

In this section, we study the problem of broadcasting in weighted trees. Beside the numerous applications of trees for designing and modeling networks, as a basic type of graphs, studying the problem in trees may be helpful for getting intuition about the nature of the problem in other classes of graphs.

The broadcasting problem in unweighted trees is already studied in [79]. From there we know that the broadcast time of an instance (T', r) of the problem (defined for unweighted tree T') can be determined in linear time using dynamic programming and working bottom-up.



(a) A uniform graph with uniform weight w

(b) A binary graph

(c) A heavy graph

(d) A light graph

(e) An even graph

(f) An odd graph

Figure 3: Weighted graph subclasses.

Here we extend the results of [79] to the weighted trees. The trees we study are considered to be undirected; however given an originator, a direction can be imagined on the edges from the originator to the leaves. This is the reason that sometime we use the term *rooted* for the trees and consider the uninformed neighbors of a node as its *children*. In this way a tree can be denoted as T_r, in which r is the root of the tree as well as the broadcast originator.

The broadcast scheme of a tree T_r can be completely described by specifying the order

in which the children of each internal node receive the message. In this way, we can represent any scheme by assigning an integer to each node $v \in V - \{r\}$, denoting the the node's index for receiving the message through its parent. We call this integer the *order number* of node v and denote it by $ord(v)$. For example in the tree of Figure 4, if the root A first informs node B, and then node C, we would have $ord(B) = 1$ and $ord(C) = 2$.

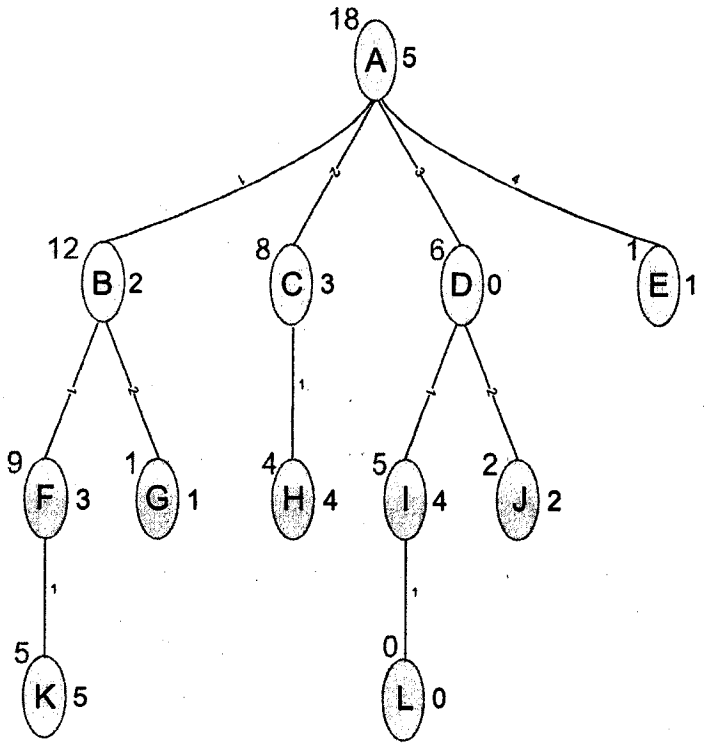


Figure 4: The broadcast scheme of the tree of Figure 2. The numbers on the right side of nodes indicate the weights, the numbers on the top left indicate the broadcast time of the subtrees, and the small numbers on the incoming links indicate the order numbers of nodes.

To determine the broadcast scheme, we calculate the broadcast time of any of the subtrees rooted at children of v. For any node v, let T_v be the subtree of T_r rooted at v (the subtree that does not contain originator). Also let $br(v, T_v)$ denote the time needed to broadcast a message from v to the nodes of T_v.

Assume that node v has p children that are labeled in a way that $br(v_1, T(v_1)) \geq$

$br(v_2, T(v_2)) \geq \ldots \geq br(v_p, T(v_p))$. Note that we can perform such ordering because working bottom-up we know the broadcast times of each T_{v_i}. Given this ordering, the broadcast time of any subtree can be calculated using the following equation:

$$br(v, T_v) = w(v) \qquad v \text{ is an external node}$$

$$br(v, T_v) = w(v) + \max_{1 \leq i \leq d} \{ br(v_i, T_{v_i}) \} + i \qquad v \text{ is an internal node}$$

Using mathematical induction, it is rather easy to verify correctness of the above recursion; we just need to consider the obvious fact that any node should first inform the subtree with the highest broadcast time. Algorithm 1 illustrates how to set broadcast times.

Algorithm 1 Tree_Set_Br_Time

Input: A weighted tree $T = (V, E)$ rooted at $r \in V$
Output: The broadcast time of any subtree T_v of T rooted at node $v \in V$. The broadcast time of T_v is stored in $br(T_v)$.

1: **if** r is an external node of T **then** {r has no children}
2: $br(T_r) = w(r)$
3: **return**
4: **end if**
5: **for** any child v of r **do**
6: Recursively call Tree_Wghtd_Br (T_v,v)
7: **end for**
8: Sort children of r in descending order of $br(T_v)$ {Hence $br(T_{v_i}) \geq br(T_{v_{i+1}})$}
9: $max = 0$
10: **for** any child v_i of r **do**
11: **if** $br(T_{v_i}) + i > max$ **then**
12: $max = br(T_{v_i} + i)$
13: **end if**
14: **end for**
15: $br(T) = w(r) + max$
16: **return**

It is not hard to verify that the dominating parameter on the complexity of the algorithm, is in line 8 of Algorithm 1, in which a node needs to sort the labels of its neighbors. Using a

comparison based sorting, the complexity of the algorithm would be $O(n \log n)$. Note that from Corollary 1, the bound $br(T_{v_i}) \leq W + n - 1$ holds for broadcast time of any of the subtrees; which enables us to improve the running time to linear by increasing the memory space. We can employ a non-comparison based sort algorithm (for example bucket sort) and spend $O(n+W)$ memory space to sort the labels in linear time. Note that if $W = O(n)$, which is the case, for example in unweighted and binary graphs, the memory space would be linear (with respect to n).

Given the broadcast times of all nodes, it is rather easy to perform broadcasting. After the time at which a vertex v completes its delay, for the next d steps it sends the message to its children in the order $v_1, v_2, \cdots v_d$ ($br(v_1, T(v_1)) \geq br(v_2, T(v_2)) \geq \ldots \geq br(v_p, T(v_p))$). The broadcast then accomplished recursively in each of the subtrees T_{v_i}. Algorithm 2 shows how to create the optimum scheme in this way.

Algorithm 2 Tree_Wghtd_Br

Input: A weighted tree $T = (V, E)$, the originator $r \in V$
Output: The optimum scheme for broadcasting from r represented as the order number of each node $v \in V - \{r\}$

1: Call Tree_Set_Br_Time(T,r) {Set the broadcast time of each subtree}
2: **for** any vertex j in the DFS traversal of T **do**
3: Let the set $Ch(j) = \{v_1, v_2, \cdots, v_d\}$ be the children set of j in ascending order of $b(v_i, T_{v_i})$.
4: **for** any $v_i \in Ch(j)$ **do**
5: Set $ord(v_i)$ to be the index of v_i in the sorted array.
6: **end for**
7: **end for**

Chapter 3

Approximating Optimum Broadcast Scheme

As mentioned in the previous chapter the broadcasting problem in weighted-vertex model is NP-Complete. In this chapter we prove the existence of algorithms which approximate the optimum broadcast scheme in arbitrary weighted-graphs. We also show there are better approximations for the problem in specific classes of weighted graphs. The approach is mainly based on reducing the broadcast problem in weighted graphs to other dissemination problems for which approximation algorithms exist.

In all reductions presented in this chapter, an instance of broadcast or multicast problem, defined on an arbitrary graph, reduces to an instance of multicast problem defined on a new graph, such that any scheme for the first instance has an equivalent scheme in the second scheme which takes the same time to complete. As the first result, the optimum dissemination time is the same in two instances. In addition, if a scheme approximates the optimum solution with ratio r in the second instance, the equivalent scheme approximates the optimum solution with the same ratio. In this way, any approximation algorithm for the second problem can be applied for the first problem as well.

We use binary graphs as an intermediate phase to reduce broadcasting in weighted-vertex graphs to multicasting in unweighted digraphs. In Section 3.1 we show that the problem of broadcasting in arbitrary weighted graphs reduces to multicasting in binary graphs. In this way we can forget about the problem in general form and focus on binary graphs. In Section 3.2 we present a reduction from multicasting in binary graphs to multicasting in directed unweighted graphs. Using the results of these two sections, in Section 3.3, we show that the problem of broadcasting in weighted undirected graphs can be reduced to multicasting in unweighted directed graphs. The later problem is already studied and there exist approximation algorithms for it, which can be applied for broadcasting in weighted graphs. In Section 3.4 we show there is a better approximation for broadcasting in heavy graphs.

3.1 Reduction from Arbitrary Weighted Graphs to Binary Weighted Graphs

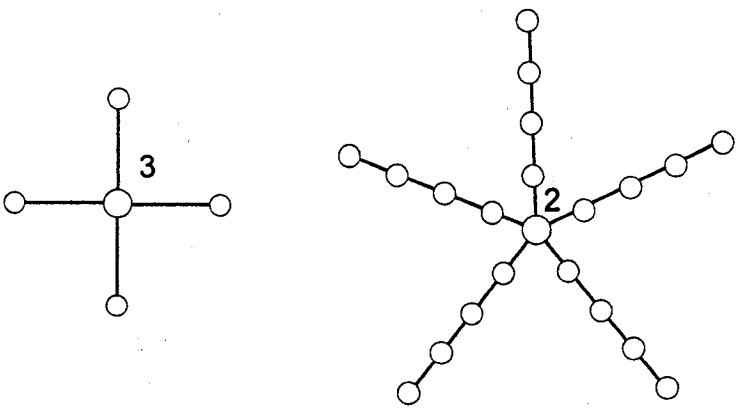
In this section we present a reduction of broadcasting problem in weighted vertex graphs to multicasting in binary graphs. More precisely we prove if there exists a scheme for broadcasting from originator u in an arbitrary weighted-graph G which completes in t rounds, then there exists a scheme for multicasting from a vertex u' to a destination set in a specific binary graph H, which needs t rounds to complete.

First, we define *insect graphs* as an instance of spider graphs presented in [5].

Definition 4. *A set of paths* $S = \{P_1, P_2, .., P_d\}$ *is called an* insect-graph *if:*
-All the paths share a vertex v *as one of their end-points.*
-All the paths are vertex-disjoint except for sharing vertex v*.*
-All paths $P_1, P_2, ..., P_d$ *have uniform length* w*.*

We call the shared vertex (v) the *head* of the insect. The other endpoints of the paths

$P_1, P_2, ..., P_d$ are called the *legs* of the insect. The *degree* of insect is the number of its legs (Consequently the size of S). The *length* of insect is the uniform length of any of the paths. A *weighted insect* is an insect in which a weight k is assigned to the head. The weights of other vertices are always considered to be 0. Figure 5 shows two samples of insect graphs.



(a) An insect of weight 3, degree 4, length 1. (b) An insect of weight 2, degree 5, length 4.

Figure 5: Two samples of insect graphs.

Theorem 8. *The problem of broadcasting in weighted graphs reduces to multicasting in binary graphs.*

Proof. Given a weighted graph G, we create a binary graph H such that any optimum broadcast scheme for G can be deduced from an optimum multicast scheme for H. To create H, we replace any vertex v of degree d in G with an insect of degree $d+1$ in a way that vertices of even weights $2k$ get replaced by insects of weight 0 and length k, and the vertices of odd weight $2k+1$ get replaced by insects of weight 1 and length k. As Figure 6 illustrates, for any edge (v, w) in G we put an edge between legs of the insects replacing v and w. Note that any leg would be connected to at most one other leg. In this way any insect would have a *hanging* leg which is not connected to any other leg.

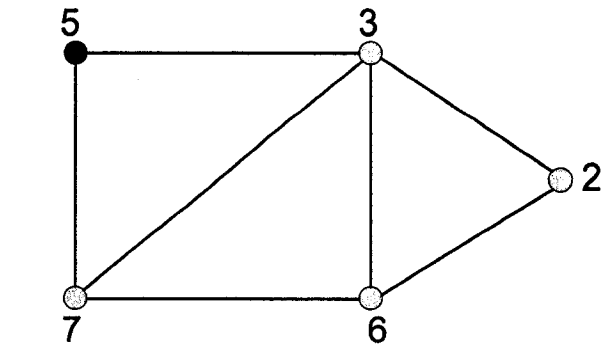
We map an instance (u, G) of the broadcast problem, to an instance $(u', H, dest(H))$

of multicast problem, in which the originator u' is the the hanging vertex of the insect replacing u, and multicast destinations ($dest(H)$) are all other hanging vertices.

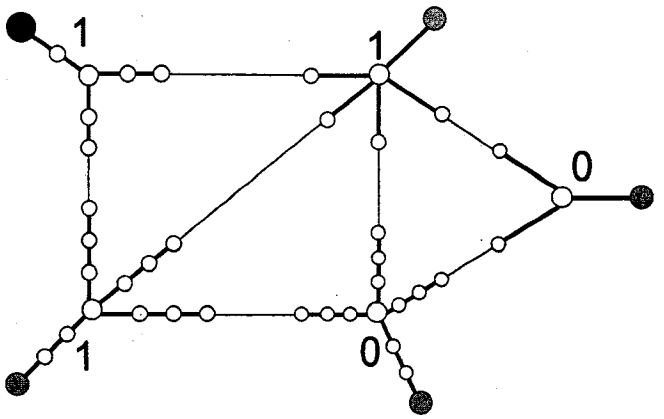
We claim that any broadcast scheme for the broadcast problem defined on G, has an equivalent multicast scheme in the multicast problem defined on H with the same time to complete. To see the proof consider an insect replacing a vertex v of weight $2k$ (or $2k+1$); this insect receives the message through one of its legs, then it takes k rounds to inform the head, 0 (or 1) rounds for the delay on the head and k more rounds to inform another leg. So the first leg receives the message after $2k$ (or $2k+1$) rounds, afterward in each round a new leg would be informed. In this way each insects replacing a vertex of weight w, can start sending the message to the neighbor insects, exactly w time units after receiving the message. Note that if a vertex of weight w is a leaf in the broadcast scheme for G, the equivalent insect needs w rounds to inform its hanging vertex. □

Theorem 9. *There exists a* $\log(n)/\log\log(n)$ *approximation for broadcasting in even graphs.*

Proof. Applying the construction of Theorem 8 on even graphs, the weights of all vertices in the resulted binary graph would be equal to 0. We conclude that any scheme for broadcasting in an even graphs has an equivalent scheme for multicasting in an unweighted graph which takes the same time to complete. Note that the size of multicast set is equal to the number of vertices in the original even graph. From this we conclude that any approximation applied for multicasting in unweighted graph can be applied for broadcasting in the even graph to get a solution with the same approximation ratio. As mentioned in Chapter 1 there exists a polynomial time algorithm which approximates the optimum multicast scheme with a ratio of $\log k/\log\log k$ in unweighted graphs in which k is the size of multicast set. We come to the conclusion that there exists a polynomial algorithm approximating the optimum broadcast scheme in even graphs with ratio $\log(n)/\log\log(n)$. □



(a) A weighted graph.



(b) The equivalent binary graph.

Figure 6: Converting a weighted graph to a binary graph. Black vertices are originators and gray vertices are multicast destinations

3.2 Reduction from Binary Weighted Graphs to Directed Unweighted Graphs

In this section we show that multicasting in undirected binary graphs reduces to multicasting in directed unweighted graphs. This reduction, together with Theorem 8, result a reduction broadcasting in undirected weighted-vertex graphs to multicasting in unweighted directed graph. This result is important in the sense that there are results for multicasting in directed graphs which can be applied for broadcasting in weighted graphs.

Like the previous section we present some auxiliary components which will be useful

in the reduction process:

Definition 5. *A* widget *from v to w is a directed cycle of length 2, 3 or 4 having v and w as two of its vertices:*

- *A* simple-widget *from v to w is simply a directed cycle having v, w as its exclusive vertices (Figure 7(a)).*
- *A* three-widget *from v to w is a directed cycle of length 3 such that the distance of v to w is 2 and distance of w to v is 1 (Figure7(b)).*
- *A* four-widget *from v to w is a directed cycle of length 4 such that the distance of v to w and w to v is 2 (Figure 7(c)).*

Note that a simple-widget from v to w is also a simple-widget from w to v. This is the same about four-widgets; however this symmetry does not hold in the case of three-widgets.

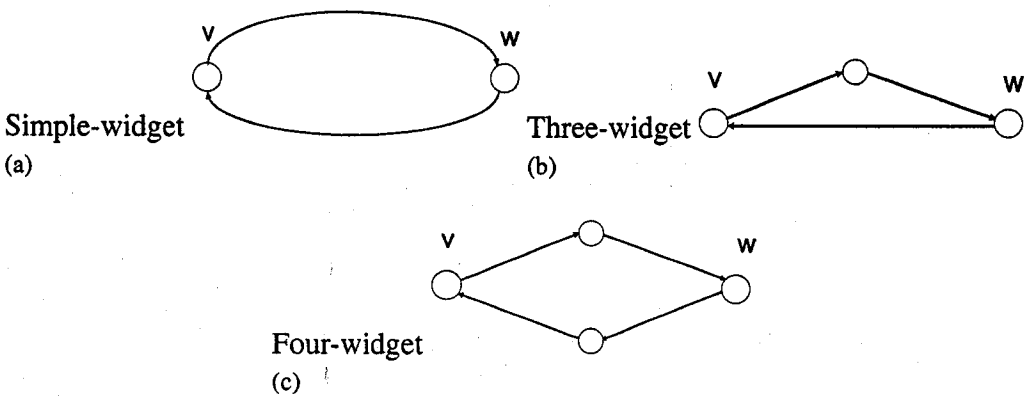


Figure 7: Widgets.

Theorem 10. *The problem of multicasting in binary graphs can be reduced to multicasting in unweighted directed graphs.*

Proof. Consider an instance of multicast problem defined on arbitrary binary graph G, with originator u, and destination set $dest(G)$. We create an instance of multicast problem for a

directed graph H.

To create H we start with the same vertex set of G. Also let the originator and multicast destinations in the new instance be the same as the instance defined on G. So what we are doing is just modifying the edges of G. As Figure 8 illustrates, we replace any edge (v, w) in G with a widget in H in the following way:

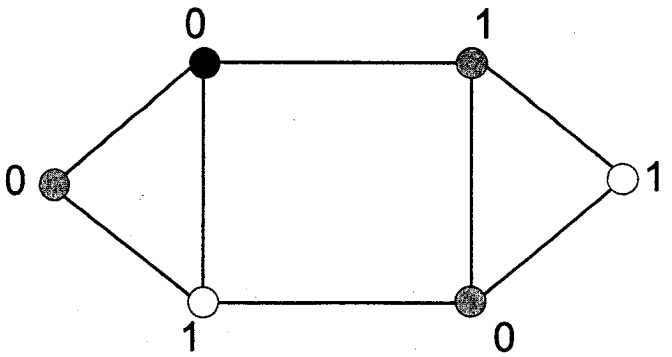
- If v and w both have weight 0, replace (v, w) with a simple-widget.
- If v has weight 1 and w has weight 0, replace (v, w) with a three-widget from v to w.
- If v and w both have weight 1, replace (v, w) with a four-widget.

We claim that any scheme for multicasting in the first instance defined on G, has an equivalent scheme in the second instance defined on H, which takes the same time to complete. To see that, consider a rooted (hence directed) tree representing a multicast scheme for G. This tree has an equivalent tree in H with the same topology in which some of the edges are replaced with paths of length 2. These are the edges having their head at a vertex of weight 1. It means that in the equivalent scheme in H, a 'one round delay' is applied to any node with weight 1 before receiving message. More precisely if in a scheme for G, a vertex w with weight 1 is informed through vertex v, in the equivalent scheme for H the edge between v and w is replaced with a path of length 2, hence the one round delay is applied in this scheme before w receives the message.

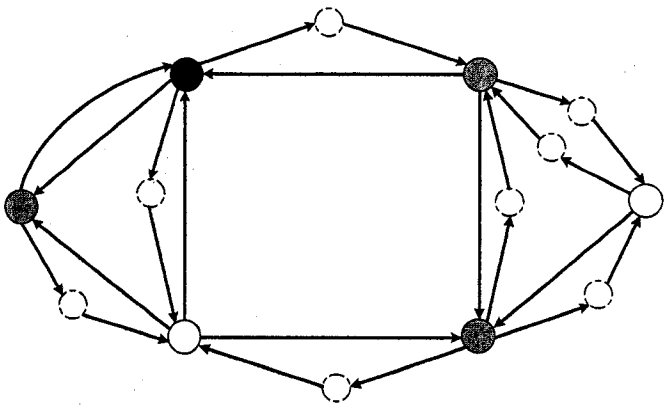
□

3.3 Approximations for Weighted Vertex-Broadcasting

The following theorem provides an approximation for k-multicast problem in directed graphs presented by Elkin and Kortsarz in [28].



(a) A binary graph.



(b) The equivalent directed unweighted graph.

Figure 8: Converting a weighted graph to a binary graph. Black vertices are originators and gray vertices are multicast destinations.

Theorem 11. *There is a polynomial time algorithm for the multicast problem in directed graphs that produces a scheme with a ratio of* $O(k^{(2/3)}/\log k)$ *of the optimum scheme in which* k *is the size of multicast set.*

The next theorem, presented by the same groupe in [27], provides a bit better approximation to the multicast problem in directed graphs:

Theorem 12. *There is a polynomial time algorithm for the multicast problem in digraphs that construct a schedule with* $O(b^* \log k + k^{1/2})$ *in which* k *is the size of multicast set and* b^* *is the optimum broadcast time.*

We can apply both Theorems 11 , 12 to come up with approximations for multicast problem in binary graphs, consequently broadcast problem in weighted graphs.

Note that the size of multicast group does not change during the reduction of Theorem 10. In other words, if we need to inform k nodes in a binary graph G, in the equivalent directed graph H also k nodes need to be informed. Applying the approximation algorithm of Theorem 11, we can come up with a scheme that approximates the optimum solution for G with ratio $O(k^{(2/3)}/\log k)$. The result of this expression is stated in the following theorem:

Theorem 13. *There is a polynomial time algorithm for the multicast problem in binary graphs that produces a scheme with a ratio of* $O(k^{(2/3)}/\log k)$ *of the optimum scheme in which* k *is the size of multicast destination set.*

The next theorem follows from Theorems 8 and 13:

Theorem 14. *There is a polynomial time algorithm for the broadcast problem in weighted-vertex graphs that produces a scheme with a ratio of* $O(n^{(2/3)}/\log n)$ *of the optimum scheme in which* n *is the number of vertices [28].*

If we apply the approximation of Theorem 12, The approximations presented in Theorems 13 and 14 enhance to the followings:

Theorem 15. *There is a polynomial time algorithm for the multicast problem in binary graphs that produces a scheme that completes in* $O(m^* \log k + k^{1/2})$ *time units in which* k *is the size of multicast destination set and* m^* *is the optimum multicast time.*

Theorem 16. *There is a polynomial time algorithm for the broadcast problem in weighted-vertex graphs that produces a scheme that completes in* $O(b^* \log n + n^{1/2})$ *time units in which* n *is the number of vertices and* b^* *is the optimum broadcast time.*

3.4 Broadcasting in Heavy Graphs

In this section we study broadcast problem in heavy graphs. We recall that heavy graphs are weighted graphs in which the weight of each node is higher than its degree. We present a reduction from broadcasting in heavy graphs to multicasting in (undirected) unweighted graphs. Applying the results on multicast problem, we deduce an approximation algorithm for the broadcast problem in heavy graphs.

Note that we do not have any NP_Completeness results for broadcasting in heavy graphs. In this sense there may be an algorithm for finding the optimum broadcast scheme in polynomial time (we intuitively believe that such an algorithm does not exist). The approximation provided here can be considered as the first solution to the problem in heavy graphs.

Theorem 17. *The problem of broadcasting in heavy graphs can be reduced to multicasting in unweighted graphs.*

Proof. Consider the broadcast problem defined on the heavy graph G with originator u. We reduce this problem to a multicast problem defined on the unweighted graph H with originator u'. We show any broadcast scheme for the first problem has an equivalent multicast scheme for the second problem which takes the same time to complete. In this way any approximation for the second problem can be applied as an approximation for the first problem with the same ratio.

To create H from G, we start with a copy of G and replace any vertex v of degree $deg(v)$ and weight $w(v)$ with a complete graph (clique) on $deg(v) + 1$ vertices in which all the links are replaced with paths of length $w(v)$. We call the vertices lying on this path *internal* nodes of the clique, while the nodes at the endpoints are *external* nodes. For each link (v, w) in G, we put an edge between two external node of the cliques replacing v and w. Each external node should be connected to at most one external node of another clique.

In this way, each clique would have exactly one *free* external node which is not connected to any other external node. Figure 9 illustrates how to construct H from G.

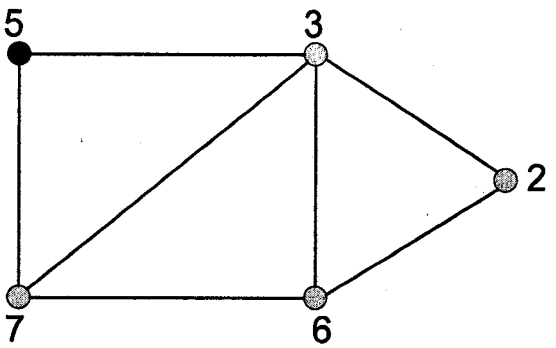
In the multicast problem defined on H, the originator would be the free vertex of the clique replacing the originator of broadcast problem defined on G, and multicast set would be the set of free vertices of all other cliques.

It can be easily verified that any scheme for the broadcast problem defined on G has an equivalent scheme for the multicast problem defined on H. To see that, consider an external node v_α of a clique receives the message through an external node of another clique. Note that v_α sends the message on all the paths toward the external nodes before any other external node of the clique receives the message. This is because the lengths of these paths is higher than the clique size. As a result, v_α informs the external vertices of its clique in time units $w, w+1, \ldots, w+deg(v)-1$ in which v is the vertex of G replaced by clique of v_α. Hence, the neighbor cliques can be informed in time units $w+1, w+2, \ldots, w+deg(v)$ which is the same as time units in which v informs its neighbors.

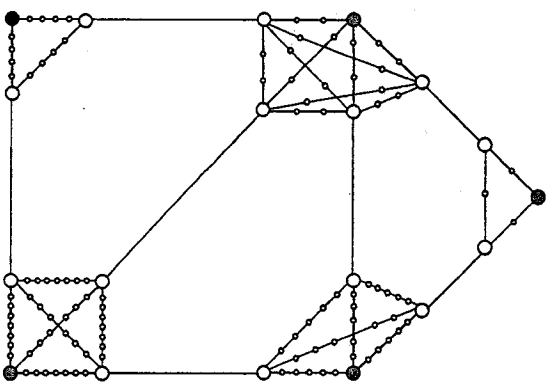
□

Corollary 3. *There exists a* $\log(n)/\log\log(n)$ *approximation for broadcasting in heavy graphs.*

Proof. From the previous theorem, we know that broadcasting in even graphs is equivalent to multicasting in unweighted graphs while the size of multicast set is equal to the size of original even graph. As mentioned before there exists a polynomial time algorithm approximating the optimum multicast scheme with a ratio of $\log k/\log\log k$ in unweighted graphs in which k is the size of multicast set. We come to the conclusion that there exists a polynomial algorithm approximating the optimum broadcast scheme in heavy graphs with ratio $\log(n)/\log\log(n)$. □



(a) A heavy graph.



(b) The equivalent unidirected graph

Figure 9: Reduction of broadcast problem in heavy graphs to multicast problem in unweighted graphs. Destination nodes are colored in gray

Chapter 4

Weighted Broadcasting in Complete Graphs

In many message-passing systems, the exact structure of the underlying communication network may not be known. In such systems it may be assumed that the network creates a complete communication graph [7]. Postal and LogP models are two well-known models in which the topology of the network is ignored and assumed to be a complete graph. In this chapter we study the weighted vertex model in complete graphs. In Section 4.1, we present an algorithm for broadcasting in any weighted complete graph in optimum time. In section 4.2, the problem is studied in uniform complete graphs, which results in tight lower and upper bounds for broadcasting in these graphs, as well as a lower bound for broadcasting in arbitrary weighted graphs.

Figure 10 shows a complete weighted-graph on 10 vertices.

4.1 Finding the Optimum Scheme in Polynomial Time

Here we look for an algorithm to find optimum broadcast scheme in weighted-vertex complete graphs. Note that in unweighted complete graphs there is a kind of symmetry which

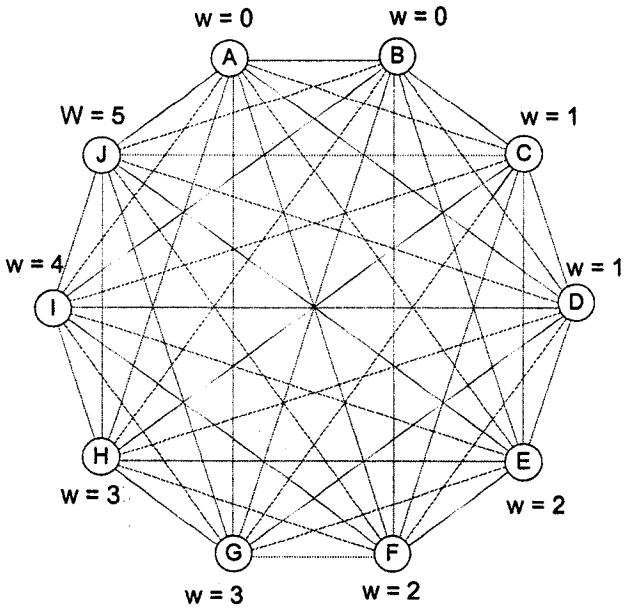
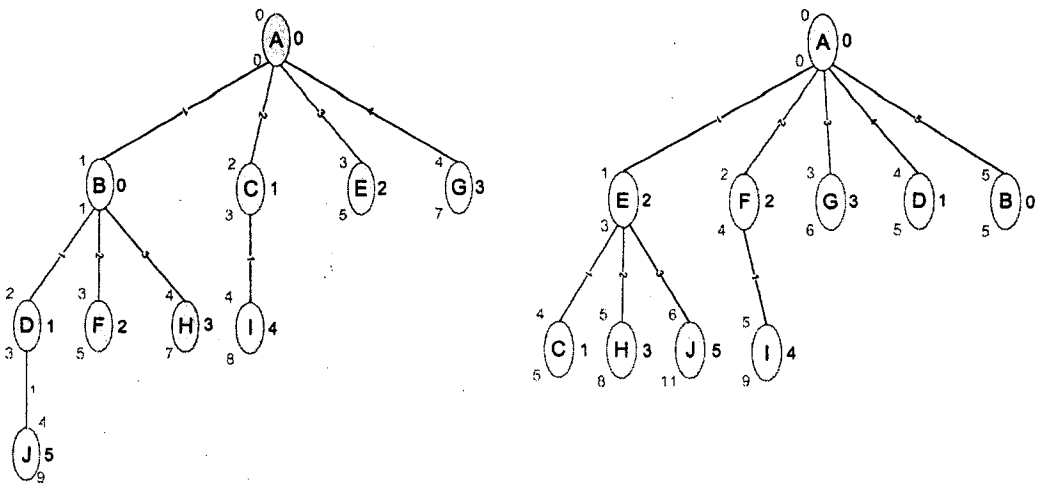


Figure 10: A complete weighted graph on 10 vertices.

makes the broadcasting trivial. In this case, any vertex after receiving a message, can arbitrarily choose one of uninformed nodes in each round to pass the message.

In complete weighted-vertex graphs the problem is not that easy in the sense that the weights assigned to the vertices are not uniform and in each round a vertex should *choose* between its neighbors based on their weights. Consider Figure 11 which shows two broadcast schemes for the graph of Figure 10, when node A is the originator. As the numbers on the nodes illustrate, the broadcast time of the first scheme is equal to 9, while the broadcast time of the second scheme is 11. It shows that a smart algorithm is needed for broadcasting in complete weighted-graphs.

To some extent, weighted-vertex model in complete graphs is similar to generalized postal model [7]. In both cases the underlying network is a complete graph and the communication model is 1-port and constant. The only difference is that in generalized postal model the weights are assigned to the edges, while in weighted-vertex model the weights



(a) A scheme in which the nodes first inform the neighbors with lower weights

(b) A scheme in which the nodes informs neighbors randomly

Figure 11: Two broadcast schemes for the complete graph of Figure 10.

are on the vertices. This small change makes the difference between the two models dramatic. With the weights on the edges, a kind of topology can be defined for graphs by disabling some edges through assigning large weights to them. In this way, the classical broadcasting reduces to broadcasting in generalized postal model. Consequently, the broadcasting problem is NP_Hard under generalized postal model. However when the weights are on the vertices, specific topologies can not be constructed by assigning large weights to the vertices; hence we can not deduce any NP_Hardness result. In contrast, we will show that there exists a polynomial algorithm for finding the optimum broadcast scheme in complete weighted-vertex graphs.

Having another look to Figure 11, it may come to mind that in a good broadcast scheme, any node first informs the neighbors with lower weights. We may conclude that in optimum broadcast trees, lower-weighted nodes construct the upper levels, while the vertices with higher weights construct the leaves and lower levels of the broadcast tree. This idea makes sense, since the neighbors with lower weights can perform their internal process faster

and participate in informing more nodes. Figure 11(a) is a broadcast tree built using this method; any node after finishing its internal process sorts uninformed neighbors based on their weights and informs theme in this order in the rest of broadcast process.

However this approach does not always create an optimum broadcast scheme. Consider a simple complete graph of Figure 12. If the originator A first informs the neighbor with lower weight, which is B, and then informs the other vertex, which is C, the broadcast time would be 11, while informing C before B leads to a broadcast time 10. So informing B before C does not improve broadcast time. The reason is that B can not inform any other node after finishing its internal process. The idea behind informing vertices with lower weights was to inform more vertices (since they finish their internal process faster) and let the broadcast tree *grow* faster. However at the end of broadcasting, most of the nodes are already informed and the internal delay of high-weight nodes is a more important issue than the growth of the broadcast tree.

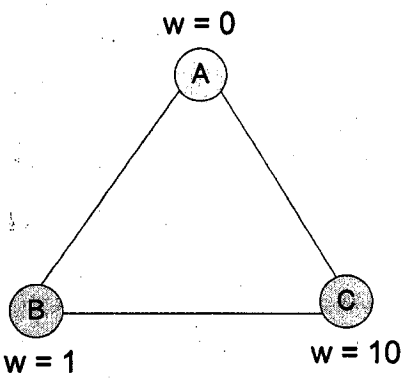


Figure 12: A simple complete graph showing that choosing the neighbor with the lowest weight does not always create the optimum scheme.

In this way, in the beginning of broadcast process (upper levels of broadcast tree), informing the low-weight vertices is effective for spreading the message faster, while at the end of broadcast process (lower levels of broadcast tree), it makes sense to inform the vertices with larger weights to make them finish their internal process faster.

We come to the conclusion that we should know the optimum broadcast time when constructing an algorithm for creating the optimum scheme. To do so, we create an algorithm that receives a *guess value* for the broadcast time. The algorithm returns either a broadcast scheme of length not greater than br or $Poor\_Guess$. We call this algorithm $broad\_Guess$ which accepts as input a weighted complete graph K_n, the originator u and the guess br for the upper bound on the optimum broadcast time for broadcasting in K_n. If the guess is correct the procedure returns a broadcast schedule for (u, K_n) of length not greater than br. In the opposite case, the procedure returns the value $Poor\_Guess$.

Note that $n - 1 + W$ is an upper bound for broadcast time in which n is the number of vertices and W is total weight of the graph (Corollary 2). In the main algorithm, we just need to conduct a binary search on the value of the guess br; on each iteration it invokes $broad\_Guess(K_n, s, br)$, the smallest value of br for which a schedule is returned is the optimum multicast time and the returned schedule would be the optimum multicast scheme.

As mentioned in Chapter 2, any broadcast scheme can be represented by a spanning tree or a sequence of calls. In the case of complete graphs, this can be easier; we represent any scheme by a sequence $< Inf(0), Inf(1), \cdots, Inf(k) >, (k \leq br)$ in which $Inf(t)$ denotes the set of nodes that get informed at time t and k is an upper bound for broadcast time. To see why such a sequence prescribes a scheme, note that in any time unit $t \leq br$ we separate the vertices into three groups:

- The set I of vertices that are informed at time t and have passed the internal delay as well (i.e. $t(v) + w(v) \leq t$). We denote them by I which stands for Informed.

- The set W of vertices that are informed, but have not passed the delay, (i.e. $t(v) + w(v) > t, t(v) \leq t$). We denote them by W which stands for Waiting.

- The set U of vertices that are not informed (i.e. $t(v) < t$). We denote them by U

standing for UnInformed.

The following fact justifies the proposed representation for the broadcast scheme:

Fact 2. *In each time unit, any multicast schedule needs to make a mapping from vertices of I to the vertices of U. Since in a complete graph any vertex of I is connected to all vertices of U this mapping is just an ordering for the vertices of U.*

To create a correct ordering of the vertices of U, in $broad\_Guess(K_n, u, br)$, in each round we separate the uninformed nodes into three groups:

- A vertex $v \in U$ is called a *late* vertex if $t + 1 + w(v) > br$ which means that v can not be informed before time unit br. Note that if there exists a late node, the multicasting can not be completed before time br. In this case the algorithm should return $PoorGuess$.

- A vertex v is called an *external* vertex if $t+1+w(v) = br$ which means that v can be informed before time unit br if and only if it gets informed at time unit t otherwise broadcasting can not be completed before time br. Note that when the number of external vertices is more than informed vertices multicasting can not complete before time unit br. In this case the algorithm again returns $Poor\_Guess$. We call these nodes external since they create the external nodes of the potential broadcast scheme.

- A vertex v is called an *internal* vertex if $t + 1 + w(v) < br$. It means that v can be informed before time unit br, and it can participate in informing the other vertices. It is rather easy to verify that if a vertex $v \in I$ have a choice between two internal vertices $v_0, v_1 \in U$, it chooses the one with smaller weight; because such a vertex can start informing the other vertices sooner . We call these nodes internal since they can be the internal nodes of the potential broadcast scheme.

Algorithm 3 broad_Guess

Input: A weighted complete graph K_n, the originator u, and a guess br for broadcast time
Output: Either a broadcast scheme that completes in less than br time units or PoorGuess

1: Initialize the sets I, W, U with Φ, $\{u\}$, $v - \{u\}$ respectively
2: Set time t to be 0
3: **while** $U \neq \Phi$ **do** {main loop on t}
4: Initialize the set $Inf(t)$ with Φ {The set of vertices that receive the message at time t}
5: **for** any vertex $v \in U$ **do**
6: **if** $t + 1 + w(v)) > br$ **then** {Late vertices}
7: **return** PoorGuess
8: **end if**
9: **if** $t + 1 + w(v)) = br$ **then** {External vertices}
10: $Inf(t) = Inf(t) \cup \{v\}$
11: $U = U - \{v\}$
12: **end if**
13: **end for**
14: k = size of I - size of $Inf(t)$
15: **if** $k < 0$ **then**
16: **return** PoorGuess {The number of external vertices is more than informed vertices}
17: **end if**
18: Extract $v_1, v_2, ...v_k \in U$ as k vertices with smallest weights in U
19: $Inf(t) = Inf(t) \cup \{v_1, v_2, .., v_k\}$
20: $U = U - Inf(t)$
21: $W = W \cup Inf(t)$
22: **for** any vertex $v \in W$ **do**
23: **if** $t(v) + w(v) = t$ **then**
24: $I = I \cup \{v\}$
25: $W = W - \{v\}$
26: **end if**
27: **end for**
28: $t = t + 1$
29: **end while**
30: **return** $< Inf(0), Inf(1), .., Inf(t) >$

The first phase of the algorithm is to initialize the sets I, W, U. At the beginning, there is no informed vertex ($I = \Phi$). While the originator is supposed to be the only waiting vertex ($W = \{s\}$), the other vertices are considered to be uninformed ($U = V - \{s\}$).

The algorithm involves a main loop that simulates the broadcasting in each round $t < br$. So an iteration of this loop is equivalent to a time unit. The goal is to create the set the vertices that get informed in each iteration ($Inf(t)$). In each time unit, first it is checked if there is any late vertex, and if the number of external vertices is more that informed vertices. In these cases the algorithm returns $PoorGuess$. Otherwise the algorithm makes an ordering of vertices of U in a way that the external vertices (if any) have the highest priority. Among the other uninformed vertices those with lower weights have higher priority because they can finish their internal process faster and contribute in message passing. Based on this ordering, the set $Inf(t)$ is constructed. Finally the sets I, W, U are updated at the end of each iteration regarding to $Inf(t)$. A precise explanation is provided in Algorithm 3.

Theorem 18. *The time complexity of the algorithm broad_Guess(K_n, u, br) is $O(n \times br)$.*

Proof. The main loop of the algorithm is performed $O(br)$ times. It is rather easy to verify that the two For loops in lines 5 and 22 of Algorithm 3 each take $O(n)$ time. Also line 18 can be performed in $O(n)$ times. To extract the $k < n$ nodes with smallest weights in linear time, we can sort the vertices which implies a preprocessing time of $O(n \log n)$ out of the main loop. Overall we come to the time complexity of $O(n \log n + n \times br)$. Applying the obvious bound $br \geq \log n$ we come to the complexity $O(n \times br)$. □

Corollary 4. *There is an algorithm with time complexity $O(n \times br \log br)$ to create the optimum broadcast scheme in an instant (u, K_n) of the broadcast problem on the complete graph K_n in which br is an upper bound for the broadcast time in K_n.*

Proof. The complexity of $broad\_Guess$ is $O(n \times br)$. As mentioned before we just need to call this algorithm $O(\log br)$ times in a binary search approach to find the smallest value of

br for which the algorithm returns a broadcast scheme. So in general the time complexity would be $O(n \times br \log br)$. □

Figure 13 shows the output of the presented approach as the optimum broadcast scheme of the complete graph of Figure 10.

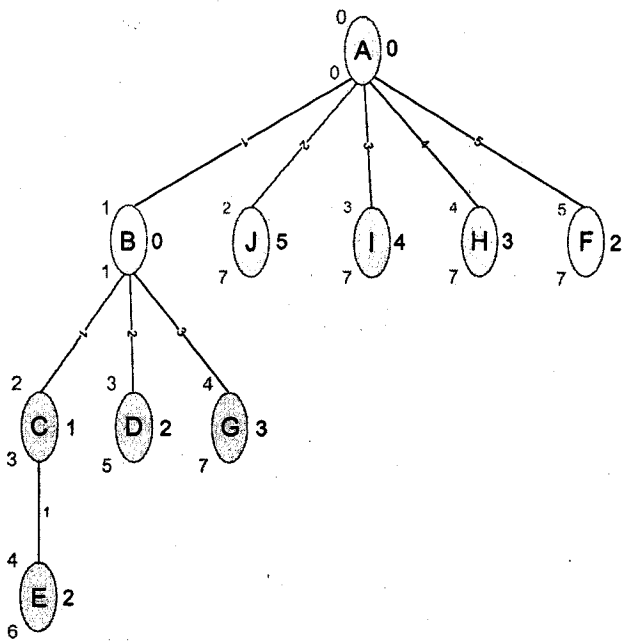


Figure 13: The optimum broadcast scheme for the complete graph of Figure 10 (node A is the originator).

4.2 Broadcasting in Uniform Complete Graphs

In this section we study the problem of broadcasting in complete graphs in which all vertices have the uniform weight w. This uniform weight can be a constant value or a function of the number on vertices. Because of the symmetric structure of the uniform complete graphs, any arbitrary none-lazy scheme can be considered as the optimum broadcast scheme. What we do in this section is to analyze the time complexity of the optimum

broadcast scheme in these graphs. This study is important because it provides a lower bound for broadcast time in uniform weighted graphs.

Theorem 19. *If $f(t)$ denotes the number of vertices that receive the message in round t in a weighted complete graph of uniform weight w then we would have:*

$$\begin{aligned} f(t) &= 1 && t = 0 \\ f(t) &= 0 && 0 < t \leq w \\ f(t) &= f(t-1) + f(t-w-1) && t > w \end{aligned}$$

Proof. It is assumed that originator receives the message at time $t = 0$, so $f(0) = 1$. Before time unit w, the originator has not completed its delay and no new vertex receive the message, hence $f(t \leq w) = 0$. At time unit w originator gets ready to inform other vertices, and at time unit $w+1$ the first neighbor of originator receives the message ($f(w+1) = 1$). In each round $t(t > w)$, any vertex that has previously informed one of its neighbors can inform another neighbor. It means the number vertices receiving the message in round t is at least equal to that of round $t-1$, which is $f(t-1)$. In addition, the vertices who have received the message at time $t-w-1$ already completed their delay at time $t-w$ and can inform one node at time t. In this way $f(t-w-1)$ additional vertices receive the message at time t. Totally we would have $f(t) = f(t-1) + f(t-w-1)$. □

Note that $f(t)$ denotes the number of vertices that receive the message just in time t. The following theorem gives the total number of vertices who have received the message:

Theorem 20. *Total number of vertices who have received the message by the end of round t, denoted by $f_{total}(t)$, is equal to $f(t+w+1)$.*

Proof. We need to show that

$$f_{total}(t) = \sum_{i=0}^{t} f(i) = f(t+w+1) \tag{1}$$

Using mathematical induction on t the proof is straightforward.

As the base of induction, it is easy to verify for any $t \leq w$, $f_{total}(t) = 1 = f(t+w+1)$. Note that for any value of t in range $w+1 \leq t \leq 2w+1$ we have $f(t) = 1$.

For any $t > w$:

$$\sum_{i=0}^{t} f(i) = \sum_{i=0}^{t-1} f(i) + f(t) = f(t+w) + f(t) = f(t+w+1)$$

The second equality is due to induction hypothesis, while the third one is based on definition of $f(t)$.

□

Note that functions $f(t)$, $f_{total}(t)$ concern the number of vertices that have received the message. However in weighted-vertex model, the broadcast time is a matter of the number of vertices that have completed their internal delay. Let $g(t)$ denote the number of vertices that complete their delay at round t. We call this number *production number* of round t.

Theorem 21. *For the production number of round t, the followings hold:*

$$g(t) = 0 \qquad t < w$$

$$g(t) = f(t-k) \qquad t \geq w$$

Proof. Note that before round w, no vertex has completed its delay. Due to uniform weights on the vertices, the vertices that complete their internal process at time $t \geq w$ are those who have received the message at time $t-w$. It means that $g(t) = f(t-w)$. □

Theorem 22. *Total number of vertices who have completed their delay at time t, denoted by $g_{total}(t)$, is equal to $f(t+1)$.*

Proof. We have to show

$$g_{total}(t) = \sum_{i=0}^{t} g(i) = f(t+1)$$

Which is rather easy to verify:

$$\sum_{i=0}^{t} g(i) = \sum_{i=w}^{t} f(i-w) = \sum_{i=0}^{t-w} f(i) = f(t+1)$$

The third equality holds based on equation (1).

□

Note that broadcasting completes when the number of vertices that have completed their delay equals to the number of vertices. It means that $b(u, K_n) = \min\{t | g_{total}(t) \geq n\}$ which results to the following theorem:

Theorem 23. *In a complete uniform graph K_n, the optimum broadcast time from any originator u can be described as $b(u, K_n) = \min\{t | f(t+1) \geq n\}$.*

Theorem 23 reveals the relevance between the function $f(t)$ and broadcast time; which makes $f(t)$ much more important to study. It should be mentioned that a similar recursion to $f(t)$ is already introduced in [7]. Also a more generalized version of the recursion is introduced in [24]. Here we present an explicit representation of $f(t)$ using an adapted version of the proof presented in [24]:

Theorem 24.

$$f(t) = \sum_{r=0}^{w} \frac{\alpha_r^{t-(w+1)}}{w + 1 - w\alpha_r^{-1}}$$

in which $\alpha_r (r = 0, 1, \ldots, w)$ are $w+1$ roots of the equation $x^{w+1} - x^w - 1 = 0$.

Proof. Note that

$$x^{w+1} - x^w - 1 = 0 \tag{2}$$

is the characteristic equation of the recursion. Hence

$$f(t) = \sum_{r=0}^{w} d_r \alpha_r^m, \tag{3}$$

in which $d_r (r = 0, 1, \ldots, (w-1))$ are arbitrary constants satisfying the recursion for $f(t)$, as can be verified by direct substitution and comparison with equation (2). So d_r must satisfy the following set of equations:

$$\begin{aligned}
f(0) &= 1 = d_0 + d_1 + \ldots + d_w \\
f(1) &= 0 = d_0\, \alpha_0 + d_1\, \alpha_1 + \ldots + d_{w-1}\, \alpha_w \\
f(2) &= 0 = d_0\, \alpha_0^2 + d_1\, \alpha_1^2 + \ldots + d_{w-1}\, \alpha_w^2 \\
&\ldots \\
f(w) &= 0 = d_0\, \alpha_0^w + d_1\, \alpha_1^w + \ldots + d_{w-1}\, \alpha_w^w
\end{aligned}$$

The Vandermonde determinant,

$$\left|\alpha_i^j\right| = \prod_{i>j} (\alpha_i - \alpha_j), \quad i, j = 0, 1, \ldots, w, \tag{4}$$

is not zero, since equation(2) has no multiple roots [24]. The determinant formed by replacing the column of $\left|\alpha_i^j\right|$ by the $f(j)$'s reduces immediately to a product involving another Vanderemonde determinant:

$$\alpha_0 \ldots \alpha_{r-1}\, \alpha_{r+1} \ldots \alpha_w \prod_{i>j} (\alpha_i - \alpha_j),$$

$$i, j = 0, 1, \ldots, (r-1), (r+1), \ldots, w.$$

By Cramer's rule, we have

$$d_r = \frac{\alpha_0 \dots \alpha_{r-1}\alpha_{r+1} \dots \alpha_w}{(-1)^{w-r}\prod(\alpha_r - \alpha_j)} = \frac{\alpha_r^w}{\prod(\alpha_r - \alpha_j)} \tag{5}$$

where $j = 0, 1, \dots, (r-1), (r+1), \dots, w$. But the denominator of the last expression is the derivative of the left member of equation (2) evaluated at $x = \alpha_r$. Hence

$$d_r = \frac{\alpha_r^w}{(w+1)\alpha_r^w - w\alpha_r^{w-1}}, \tag{6}$$

and, therefore,

$$f(t) = \sum_{r=0}^{w} \frac{\alpha_r^{(t-(w+1))}}{w + 1 - w\alpha_r^{-1}}$$

which completes the proof. □

The previous theorem along with Theorem 23 result in the following theorem:

Theorem 25. *In the complete uniform graphs, if the weights on the vertices be a constant value, the broadcasting completes in* $\Theta(\log n)$.

Proof. Let $p(t) = t^{w+1} - t^w - 1$ be the characteristic polynomial of $f(t)$. Note that $p(t)$ is a continuous function and $p(1) = -1 < 0$, $p(2) > 1$. Applying intermediate value theorem [80], we deduce that $p(t)$ has a real root α_x in the interval $(1, 2)$. As a result we can say:

$$f(t) \geq \frac{\alpha_x^{(t-(w+1))}}{w + 1 - w\alpha_x^{-1}} = c\alpha_x^{t-c'}$$

in which c, c' are constant values. So $f(t) = \Omega(\alpha_x^t)$. Applying this bound on Theorem 23 we would have $br(u, K_n) = O(\log n)$

In addition, it is easy to verify that $f(t)$ decreases with respect to w, which means

$$f_w(t) < f_{w-1}(t) < \dots < f_0(t)$$

Applying equation $f_0(t) = 2^t$, we would have $f(t) = O(2^t)$. As a result $f(t) = O(2^t)$. Again using Theorem 23 we can conclude $br(u, K_n) = \Omega(\log n)$.

Having $br(u, K_n) = O(\log n)$, and $br(u, K_n) = \Omega(\log n)$ we conclude $br(u, K_n) = \Theta(\log n)$.

□

We can use the results obtained for complete graphs to present a lower bound for broadcast time in weighted graphs in general sense.

Corollary 5. *For any originator u in arbitrary weighted graph G, we have $br(u, G) = \Omega(\log n)$ in which n is the number of vertices in G.*

Proof. Given a weighted graph G, we create a complete weighted graph K_n from G by adding all missing edges of G to K_n. These extra edges reduce broadcast time of K_n, so $br(u, K_n) \leq br(u, G)$. Now if we reduce the weights of vertices in K_n to w_{min}, in which w_{min} is the minimum weigh on a vertex of G, we come up with a uniform weighted graph K'_n such that $br(u, K') \leq br(u, K)$. From Theorem 25 we know that $br(u, K') = \Theta(\log n)$. Applying $br(u, G) \geq br(u, K)$ we conclude $br(u, G) = \Omega(\log_\varphi n)$.

□

Note that when the uniform weight of vertices is equal to 1, the recursion of $f(t)$ turns out to be the well-known Fibonacci sequence. In this case we have $g_{total}(t) = f(t+1) = \left\lfloor \frac{\Phi^{(t+1)}}{\sqrt{5}} + .5 \right\rfloor$ in which Φ is the *Golden Ratio* constant [17]. Applying Theorem 4.2, we would have:

Theorem 26. *In any complete weighted graph K_n in which all vertices have uniform weight $w = 1$, for any originator u we have $b(u, K_n) = \log_\Phi(n-1)$ in which n is the number of vertices and Φ is the golden ratio constant.*

Chapter 5

Heuristic Algorithms for Weighted Vertex Broadcasting

In this chapter we present three heuristic algorithms for the broadcast problem in weighted vertex graphs. In Section 5.1, a modified version of Dijkstra's algorithm is presented which creates weighted shortest path trees as potential good answers to the problem. In Section 5.2, a greedy algorithm is described; while in Section 5.3, an evolutionary algorithm is proposed for the problem. In Section 5.4, the performance of these methods is compared using computer simulation. It should be mentioned that in designing these algorithms we used the idea behind the algorithms presented in [8]. There, the authors were interested in heuristics for broadcasting in complete graphs under generalized postal model.

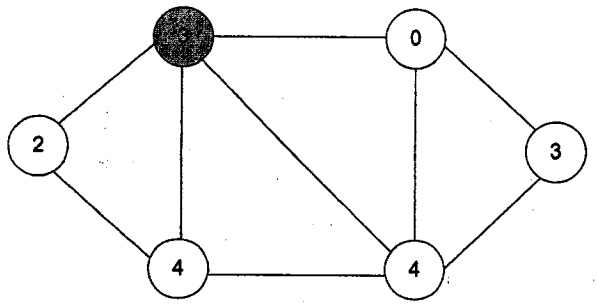
5.1 Modified Dijkstra's Algorithm

Dijktra's algorithm solves the single-source shortest-path problem in an arbitrary graph. A version of this algorithm for creating shortest path trees in weighted-vertex graphs is studied in [9]. The cost of a path between two vertices is considered to be the length of the path plus the total weights of the vertices on that path. Note that originator has a cost

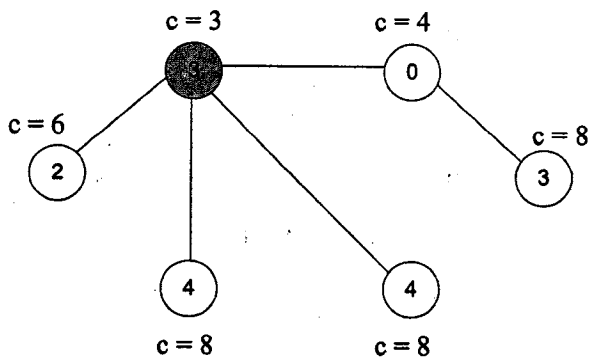
as well, which is equal to its weight. Figure 14(a) shows a weighted vertex graph; and Figure14(b) denotes the shortest path tree created by Dijkstra's algorithm. Note that $c(v)$ denotes the cost of a node v in the shortest path tree.

Here we apply Dijkstra's algorithm to solve the broadcast problem in weighted-vertex graphs. Having originator v_0 as the source of the tree, the shortest path tree rooted at v_0 can be assumed to be a good solution to the problem.

The algorithm is the same as classical Dijkstra's algorithm. The only difference is that while adding a node v to the output tree, the weight of v is also added to the cost of v. Algorithm 4 is a pseudocode for the weighted vertex Dijkstra's algorithm. To present the output tree, the algorithm determines the parent of each node (except originator) in the tree.



(a) A weighted weight graph



(b) The output tree of Dijkstra's algorithm

Figure 14: Applying modified Dijkstra's algorithm for the weighted-vertex broadcasting. The dark node is the originator.

In [9], it is proved that, for any weighted vertex graph $G(V, E)$, the time complexity of

Algorithm 4 Dijkstra's Algorithm

Input: A weighted graph $G = (V, E)$, the originator $v_0 \in V$
Output: The shortest path tree from v_0 as a scheme for broadcasting from v_0 in G.

1: **for** any vertex $v_i \in V$ **do**
2: $c(v_i) \leftarrow \infty$
3: $p(v_i) \leftarrow NULL$
4: **end for**
5: $I \leftarrow \{v_0\}$ {the set of Informed vertices}
6: $c(v_0) \leftarrow w(v_0)$
7: $U \leftarrow V - \{v_0\}$ {the set of Uninformed vertices}
8: **while** $U \neq \phi$ **do**
9: Find $u \in I, v \in U$, $(u, v) \in E$ such that
$c(u) + w(v) = \min_{i,j}(c(i) + w(j))\ [i \in I, j \in U, (i, j) \in E]$
10: $p(v) \leftarrow u$
11: $c(v) = c(u) + w(v) + 1$
12: $I \leftarrow I \cup \{v\}$
13: $U \leftarrow U - \{v\}$
14: **end while**
15: **return** $p(v_0), p(v_1), \cdots p(v_n)$

Dijkstra's algorithm, is $\mathcal{O}(|E| + |V| \log |V|)$.

5.2 The Greedy Algorithm

In Dijkstra's algorithm, the cost of a node remains constant once it has been set, ignoring the fact that sending message to each neighbor requires a separate unit of time. Here we present a greedy algorithm in which this issue is considered. The algorithm is based on the modified Dijkstra's algorithm. Let I represents the set of informed vertices in each iteration of the algorithm, while U is the set of uninformed vertices. When a link (u, v) $[u \in I, v \in U]$ is attached to the broadcast tree, after setting the cost of w, the cost of u increases one unit. As a result, an extra cost would be applied in the cost of any other neighbor of u which is going to be informed in future iterations. Algorithm 5 is a pseudocode for this approach.

Note that the only difference between Algorithm 5 and Algorithm 4 is the additional step in line 12 of Algorithm 5. However this *small* difference dramatically enhances greedy approach in comparison with Dijkstra's algorithm (as will be discussed later). Figure 15

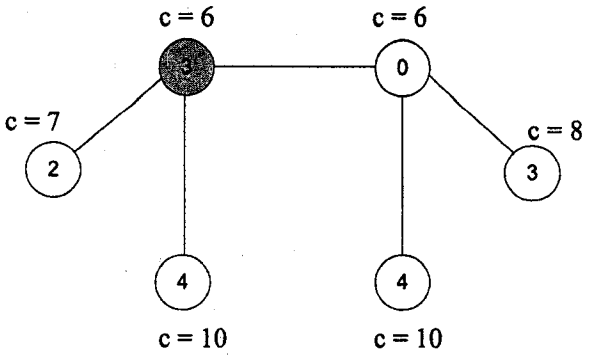


Figure 15: Applying greedy algorithm on the graph of Figure 14(a). The dark node is the originator.

shows the output by the greedy method for the graph of Figure 14(a). It is easy to verify that the broadcast time of this tree is equal to 9. Note that the broadcast time of the tree generated by modified Dijkstra's algorithm (Figure14(b)) is 10. Later we generalize this result to state that the performance of greedy algorithm, in average, is much better than Dijkstra's algorithm.

5.3 The Evolutionary Algorithm

There are another type of heuristic algorithms that can be applied for the broadcast problem based on evolutionary processes. In [59] a genetic algorithm is proposed for the problem in classical model. Also a simulated annealing approach is presented in [8] for broadcasting in generalized postal model. Here we introduce an evolutionary algorithm for broadcasting in weighted-vertex graphs. The algorithm is desired to be simple enough for future theoretical analysis.

To obtain a good scheme for an instance $G = (u, V)$ of broadcasting problem, We start with a candidate solution (broadcast tree) which is initialized randomly. Then the solution is improved in an evolutionary process. In each iteration of the evolution, a small change, called *mutation*, is applied on the tree. If the broadcast time of the mutated tree is

Algorithm 5 Greedy Algorithm

Input: A weighted graph $G = (V, E)$, the originator $v_0 \in V$
Output: A tree rooted at v_0 as a scheme for broadcasting from v_0 in G.

1: **for** any vertex $v_i \in V$ **do**
2: $t(v_i) \leftarrow \infty$
3: $p(v_i) \leftarrow NULL$
4: **end for**
5: $I \leftarrow \{v_0\}$ {the set of Informed vertices}
6: $c(v_0) \leftarrow w(v_0)$
7: $U \leftarrow V - \{v_0\}$ {the set of Uninformed vertices}
8: **while** $U \neq \phi$ **do**
9: Find $u \in I, v \in U$, $(u, v) \in E$ such that
$c(u) + w(v) = \min_{i,j}(c(i) + w(j))\ [i \in I, j \in U, (i, j) \in E]$
10: $p(v) \leftarrow u$
11: $c(v) = c(u) + w(v) + 1$
12: $c(u) = c(u) + 1$
13: $I \leftarrow I \cup \{v\}$
14: $U \leftarrow U - \{v\}$
15: **end while**
16: **return** $p(v_0), p(v_1), \cdots p(v_n)$

better than the original one, the new tree will be *accepted*; otherwise the algorithm *rejects* it. This process continues up to the point that the broadcast times of mutated trees do not improve after a series of iterations. In this situation we say the system is *freezed* and stop the evolution process.

In this way, the process starts at a random solution and gradually moves to better and better states until it cannot improve any more. Algorithm 6 precisely describes the evolution process. The algorithm works based on the following assumptions:

- A mutation is applied by removing a random edge of the tree, thus splitting a subtree off it. Then the subtree is rejoined to the main tree by inserting an edge between the root of subtree and a random node of the main tree.

- The algorithm always keeps the *best solution* to returns it, if at the end of the process no better solution was found.

- An *Equilibrium* is reached if no improvement occurs after a certain number of iterations (In Algorithm 6 this number is stored in valuable *ctr*). In this situation we reset the *current solution* with the preserved *best solution.*

- After a certain number of repeated equilibriums, the system is considered to be freezed and the algorithm terminates (in Algorithm 6 this number is stored in valuable $freezLevel$).

Algorithm 6 Evolutionary Algorithm

Input: A weighted graph $G = (V, E)$, the originator $v_0 \in V$, $FreezBond$ [the freezing bound indicating termination condition].
Output: A tree rooted at v_0 as a scheme for broadcasting from v_0 in G.

1: $curTree \leftarrow$ a random schedule (spanning tree) of G {the current solution that is evolving }
2: $bestTree \leftarrow curTree$, $bestTime \leftarrow br_{v_0}(curTree)$ {the best solution to the problem is kept in these variables}
3: $freezLevel = 0$
4: $cnt = 0$
5: **while** $freezLevel < freezBond$ **do**
6: $curTree \leftarrow mutation(curTree)$
7: **if** $br_{v_0}(curTree) < bestTime$ **then**
8: $bestTree \leftarrow curTree$
9: $bestTime = br_{v_0}(curTree)$
10: $freezLevel = 0$
11: **else**
12: $cnt = cnt + 1$
13: **if** $cnt > freezBond$ **then**
14: $curTree \leftarrow bestTree$
15: $freezLevel = freezLevel + 1$
16: **end if**
17: **end if**
18: **end while**
19: **return** $bestTree$

Note that the time complexity of the algorithm is a matter of $freezBond$. Higher values for this value may result in better solutions, however it may dramatically increase the time complexity of the algorithm.

5.4 Simulations and Analysis

In this section we compare the performance of the heuristics introduced in the previous section based on computer simulation. Note that theoretically we can not say that any of the heuristics always performs better than another one. It may come to mind that the output by the greedy algorithm is always better than modified Dijkstra's algorithm. However it is not true. Figure 16 shows a graph and output trees of the two algorithms. It can be verified that broadcasting in the scheme generated by Dijkstra's algorithm completes after 4 rounds; while the scheme created by greedy method needs 5 rounds to complete.

Here, we compare the average performance of the algorithms using computer simulation. We implemented the algorithms presented in the previous sections, as well as an algorithm for creating random spanning trees as the baseline solution. We applied each algorithm 50 times on 100 different randomly generated graphs.

While creating random graphs, it is assumed that a specific number of edges is distributed uniformly between nodes of graph. Also it is assumed that the weights assigned to the nodes obey a normal distribution.

Here, it is studied how the average weight and the average degree of vertices affect the efficiency of the algorithms.

- **Average weight of vertices:**

  In this experiment, we create random graphs on $n = 200$ vertices and 4000 edges (so the average degree of each node is 40) to study the performance of the algorithms when the average weight of the vertices changes. Note that the topology of graphs is preserved when the weights change. Table 1 contains the broadcast times of the algorithms in each cases.

  As expected, in average, the results of greedy algorithm are much better than Dijkstra's algorithm. Also, we can see when the weights grow up, the gap between the

$\bar{w}$	*Dijkstra*	*Greedy*	*Random*	*Evolutionary*
0	42	8	12	11
1	46	9	21	17
2	53	13	29	26
5	53	25	75	62
10	60	40	121	107
20	69	70	255	214
40	154	157	573	457
60	230	234	766	690
80	266	267	1034	791
100	340	339	1188	999

Table 1: The broadcast time of the heuristics, when the average weight of vertices ($\bar{w}$) changes; The number of vertices is 200 and the number of edges is 4000.

performance of the greedy algorithm and Dijkstra's algorithm decreases. A simple explanation is that when the weights increase, this is the weight (and not degree) of the vertices lying on the paths from originator to the leaves, that mainly determines the broadcast time; while the the strength of the greedy algorithm is in producing schemes with lower degrees. Note that the performance of evolutionary algorithm decreases when the weights grow up. We should recall that the efficiency of the evolutionary algorithms is a matter of *freezBound*. In this experiment, this parameter is set to be 10 which is relatively low. However, as Table 1 shows, this 'fast' algorithm evolves randomly made trees in a reasonable rate.

- **Average degree of vertices:**

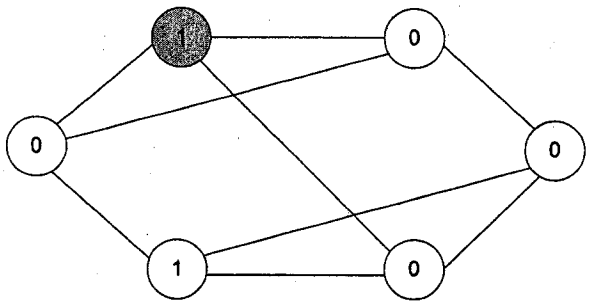
  In this experiment, we create graphs on $n = 200$ vertices and average weight of $\bar{w} = 10$. We change the number of edges to study the performance of the algorithms when the average degree of nodes changes. Table 1 contains the broadcast times of the algorithms in each case.

  Theoretically, the broadcast time of the optimum scheme improves when the average

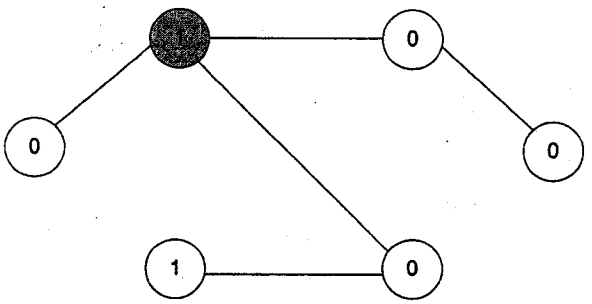
$\bar{deg}$	*Dijkstra*	*Greedy*	*Random*	*Evolutionary*
30	72	55	140	133
40	82	46	135	131
50	78	50	135	128
60	83	44	133	125
70	82	51	131	123
80	92	45	128	123
90	103	46	122	115
100	193	44	121	114
150	196	43	107	102

Table 2: The broadcast time of the heuristics, when the average degree of vertices ($\bar{deg}$) changes; The number of vertices is assumed to be 200 and the average weight is 10.

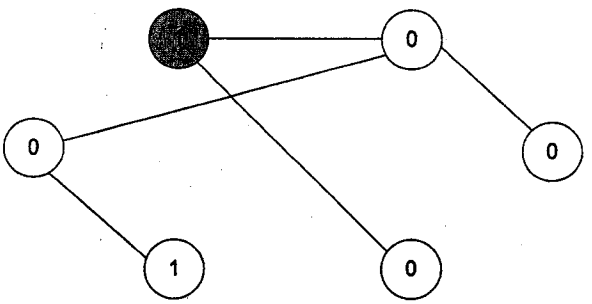
degree increases. However in Table 2 it can be seen that the performance of Dijkstra's algorithm decreases when the number of edges grows. The reason is that in this algorithm, the cost caused by the degree of the vertices does not come to account; the output of Dijkstra's algorithm in large-degree graphs are trees with high-degree and small diameter. As an extreme example, in complete graphs, the output of Dijkstra's algorithm is a star tree; which is the worst possible scheme (much wost than a random scheme). The greedy approach is also based on minimizing the length of paths. So in dense graphs the algorithm tends to create trees with small diameter. However the cost applied for extra children compromises this drawback to some extent. In the case of random trees and trees generated by evolutionary algorithms, the broadcast time reduces in high-degree graphs; which is quite natural.



(a) A weighted graph



(b) The output tree of Dijkstra algorithm



(c) The output tree of greedy algorithm

Figure 16: A graph in which modified Dijkstra's algorithm performs better ($br = 4$) than the greedy algorithm ($br = 5$).

Chapter 6

Conclusion and Future Work

In this thesis we presented a new method for information dissemination in which the underlying communication network is modeled by a weighted-vertex graph. As a generalization of the unweighted graphs, broadcasting in weighted graphs is more tricky than unweighted graphs. While the broadcasting problem is NP_Complete for the weighted graphs, the problem remains NP_Hard for some classes of the weighted graphs including uniform graphs, light graphs, even graphs and binary graphs. Verifying NP_Completeness for uniform and odd graphs is not easy, and is a subject for future work. We show the existence of approximation algorithms for the problem in general term, as well as better approximations for some subclasses of weighted graphs. Also we studied the broadcasting problem in complete weighted graphs, and presented a polynomial algorithm for finding the optimum scheme in those graphs. Specifically, we discussed the broadcast time of complete graphs with uniform weights on the vertices. Finally, we presented three heuristics for the problem, and using computer simulation compared their performance.

As the future work, I am mainly interested in studying the problem when the weights assigned to vertices may be negative. Although negative weights do not make sense in most of the applications, this taste of problem seems to have interesting characteristics in theoretical aspects.

Another interesting problem is broadcasting in weighted digraphs. All the weighted

graphs discussed in this thesis were assumed to be undirected. In the case of weighted digraphs, the problem turns to be a different (and much harder) problem that should be studied separately.

Creating weighted graphs with good performance under specific limitation on the number of links and weights is another subject that may have applications in real world. This would generalize the efforts performed for creating graphs on n vertices with minimum number of edges in which broadcasting can be completed within theoretically minimal time.

What we did here was actually applying classical model (telephone model) on weighted graphs. Broadcasting in weighted graphs under other existing models is another subject for future work. Most of the models defined in Chapter 1 can be applied for weighted vertex graphs as well.

Classical model can be applied in the weighted graphs in a different way to achieve a new model. In this approach, it may be considered that in the broadcast trees the costs of leaves not be added to the broadcast time. In this way, broadcasting completes when all leaves 'receive' the message (they don't need to perform internal delay). This assumption, dramatically changes the model and most of the results of this thesis can not be applied on that model. Studying this model may be a subject for future work.

Bibliography

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

[2] B. Aiello, F. T. Leighton, B. M. Maggs, and M. Newman. Fast algorithms for bit-serial routing on a hypercube. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 55–64, 1990.

[3] T. Asaka, T. Miyoshi, and Y. Tanaka. Multicast routing in satellite-terrestrial networks. *Fifth Asia-Pacific Conference on Communications and Fourth Optoelectronics and Communications Conference*, 1:768–771 vol.1, 1999.

[4] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Multicasting in heterogeneous networks. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 448–453, New York, NY, USA, 1998. ACM.

[5] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Multicasting in heterogeneous networks. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 448–453, New York, NY, USA, 1998. ACM.

[6] A. Bar-Noy and C-T. Ho. Broadcasting multiple messages in the multiport model. *IEEE Transactions on Parallel and Distributed Systems*, 10(5):500–508, 1999.

[7] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 13–22, 1992.

[8] A. Bar-Noy and U. Nir. The generalized postal model-broadcasting in a system with non-homogeneous delays. *Electrotechnical Conference, 1998. MELECON 98., 9th Mediterranean*, 2:1323–1327 vol.2, 18-20 May 1998.

[9] Michael Barbehenn. A note on the complexity of dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on Computers*, 47(2):263, 1998.

[10] B. Beauquier, S. Pérennes, and O. Delmas. Tight bounds for broadcasting in the linear cost model. *Journal of Interconnection Networks*, 2(2):175–188, 2001.

[11] A. Bellaachia and A. Youssef. Personalized broadcasting in banyan-hypercube networks. *Computer Communications and Networks, 1995. Proceedings., Fourth International Conference on*, pages 470–474, 20-23 Sep 1995.

[12] J-C. Bermond, H. A. Harutyunyan, A. L. Liestman, and S. Perennes. A note on the dimensionality of modified knödel graphs. *International Journal of Foundations of Computer Science*, 8(2):109+, 1997.

[13] J-C. Bermond, P. Hell, A. L. Liestman, and J. G. Peters. Broadcasting in bounded degree graphs. *SIAM J. Discret. Math.*, 5(1):10–24, 1992.

[14] J-C. Bermond, P. Hell, A. L. Liestman, and J. G. Peters. Sparse broadcast graphs. *Discrete Appl. Math.*, 36(2):97–130, 1992.

[15] P. Berthome, A. Ferreira, and S. Perennes. Optimal information dissemination in star and pancake networks. *IEEE Transactions on Parallel and Distributed Systems*, 07(12):1292–1300, 1996.

[16] B. Birchler, A. Esfahanian, and E. Torng. Information dissemination in restricted routing networks. In *International Symposium on Combinatorics and Applications*, pages 33–44, 1996.

[17] D. M. Burton. *The History of Mathematics : An Introduction.* McGraw-Hill Science/Engineering/Math, 2002.

[18] F. Comellas and C. Dalfo. Optimal broadcasting in 2-dimensional manhattan street networks. volume 246, pages 135–140. Utilitas Mathematica Publishing Inc, 2005.

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

[20] F. Comellas and P. Hell. Broadcasting in generalized chordal rings. *Networks*, 42(3):123134, 2003.

[21] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. Von Eicken. Logp: towards a realistic model of parallel computation. *SIGPLAN Not.*, 28(7):1–12, 1993.

[22] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *Computers, IEEE Transactions on*, C-36(5):547–553, May 1987.

[23] A. Dessmark, A. Lingas, H. Olsson, and H. Yamamoto. Optimal broadcasting in almost trees and partial k-trees. In *STACS '98: Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science*, pages 432–443, London, UK, 1998. Springer-Verlag.

[24] D. Dickinson. On sums involving binomial coefficients. *The American Mathematical Monthly*, 57(2):82–86, 1950.

[25] M. J. Dinneen, M. R. Fellows, and V. Faber. Algebraic constructions of efficient broadcast networks. In *AAECC-9: Proceedings of the 9th International Symposium, on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 152–158, London, UK, 1991. Springer-Verlag.

[26] M. Elkin and G. Kortsarz. Combinatorial logarithmic approximation algorithm for directed telephone broadcast problem. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 438–447, New York, NY, USA, 2002. ACM.

[27] M. Elkin and G. Kortsarz. Approximation algorithm for directed telephone multicast problem. *Lecture Notes in Computer Science*, 2719:188, 2003.

[28] M. Elkin and G. Kortsarz. Sublogarithmic approximation for telephone multicast: path out of jungle (extended abstract). In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 76–85, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[29] M. Elkin and G. Kortsarz. Logarithmic inapproximability of the radio broadcast problem. *J. Algorithms*, 52(1):8–25, 2004.

[30] S. Even and B. Monien. On the number of rounds necessary to disseminate information. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 318–327, 1989.

[31] A. M. Farley. Minimal broadcast networks. *Networks*, 9(4):313–332, 1978.

[32] A. M. Farley. Minimum-time line broadcast networks. *Networks*, 10(1):59–70, 1980.

[33] A. M. Farley and S. T. Hedetniemi. Broadcasting in grid graphs. In *the* 9^{th} *Conf. Combinatorics, graph theory, and computing*, pages 275–288. Utilitas Mathematica Publishing Inc, 1978.

[34] 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.

[35] P. Fraigniaud. Approximation algorithms for collective communications with limited link and node-contention. Technical Report LRI-1264, Universite Paris-Sud, France, 2000.

[36] P. Fraigniaud. Approximation algorithms for minimum-time broadcast under the vertex-disjoint paths mode. *Lecture Notes in Computer Science*, 2161:440+, 2001.

[37] P. Fraigniaud. Minimum-time broadcast under edge-disjoint paths modes. In *International conference on fun with algorithm*, 2001.

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

[39] P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks. *Discrete Appl. Math.*, 53(1-3):79–133, 1994.

[40] P. Fraigniaud and S. Vial. Approximation algorithms for information dissemination problems. *IEEE Second International Conference on Algorithms and Architectures for Parallel Processing (ICAPP)*, pages 155–162, 1996.

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

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

[43] T. Hart and H. A. Harutyunyan. Improved messy broadcasting in hypercubes and and complete bipartite graphs. *Congressus Numerantium*, 156:124–140, 2002.

[44] T. Hart and H. A. Harutyunyan. New bounds on messy broadcasting. *Conressus Numerantium*, 2002.

[45] H. A. Harutyunyan. Multiple message broadcasting in modified knödel graph. In *SIROCCO*, pages 157–165, 2000.

[46] H. A. Harutyunyan. Minimum multiple message broadcast graphs. *Networks*, 47(4):218–224, 2006.

[47] H. A. Harutyunyan and A. L. Liestman. More broadcast graphs. *Discrete Applied Mathmatics*, 98(1-2):81–102, 1999.

[48] H. A. Harutyunyan and A. L. Liestman. Improved upper and lower bounds for k-broadcasting. *Networks*, 37(2):94–101, 2001.

[49] H. A. Harutyunyan and A. L. Liestman. k-broadcasting in trees. *Networks*, 38(3):163–168, 2001.

[50] H. A. Harutyunyan and A. L. Liestman. On the monotonicity of the broadcast function. *Discrete Math*, 262(1-3):149–157, 2003.

[51] H. A. Harutyunyan, A.L. Liestman, and B. Shao. A linear algorithm for finding the k-broadcast center of a tree. *Networks*, to appear.

[52] H. A. Harutyunyan and E. Maraachlian. Linear algorithm for broadcasting in unicyclic graphs. In *International Computing and Combinatorics Conference (COCOON)*, pages 372–382, 2007.

[53] H. A. Harutyunyan and E. Maraachlian. On broadcasting in unicyclic graphs. *Journal of Combinatorial Optimization (JCO)*, 2008 (to appear).

[54] H. A. Harutyunyan and B. Shao. An efficient heuristic for broadcasting in networks. *Journal of Parallel and Distributed Computing*, 66(1):68–76, 2006.

[55] H.A. Harutyunyan and A.L. Liestman. Messy broadcasting. *Parallel Process. Lett.*, 8:149–159, 1998.

[56] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–359, 1998.

[57] P. Hell and K. Seyffarth. Broadcasting in planar graphs. *Australas. J. Combin.*, 17:309318, 1998.

[58] M-C. Heydemann, J. Opatrny, and D. Sotteau. Broadcasting and spanning trees in de bruijn and kautz networks. *Discrete Applied Math*, 27-28:297–317, 1992.

[59] Cory J. Hoelting, Dale A. Schoenefeld, and Roger L. Wainwright. A genetic algorithm for the minimum broadcast time problem using a global precedence vector. In *SAC '96: Proceedings of the 1996 ACM symposium on Applied Computing*, pages 258–262, New York, NY, USA, 1996. ACM.

[60] J. Hromkovic, R. Klasing, B. Monien, and R. Peine. Dissemination of information in interconnection networks (broadcasting & gossiping). *Combinatorial Network Theory*, pages 125–212, 1996.

[61] J. Hromkovic, R. Klasing, and E. Stohr. Dissemination of information in vertex-disjoint paths mode, part 1: General bounds and gossiping in hypercube-like networks. In *Information and Computation*, 1993.

[62] J. Hromkovic, R. Klasing, W. Unger, and H. Wagener. Optimal algorithms for broadcast and gossip in the edge-disjoint path modes (extended abstract). In *Scandinavian Workshop on Algorithm Theory*, pages 219–230, 1994.

[63] A. Jakoby, R. Reischuk, and C. Schindelhauer. The complexity of broadcasting in planar and decomposable graphs. *Discrete Appl. Math.*, 83(1-3):179–206, 1998.

[64] K. Jansen and H. Müller. The minimum broadcast time problem for several processor networks. *Theoretical Computer Science*, 147(1–2):69–85, 1995.

[65] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[66] L. H. Khachatrian and H. S. Haroutunian. Construction of new classes of minimal broadcast networks. In *Proceedings 3rd International Colloquium on Coding Theory*, pages 69–77, 1990.

[67] J-C. König and E. Lazard. Minimum k-broadcast graphs. *Discrete Applied Mathematics (DAM)*, 53(1-3):199–209, 1994.

[68] G. Kortsarz and D. Peleg. Approximation algorithms for minimum-time broadcast. *SIAM J. Discret. Math.*, 8(3):401–427, 1995.

[69] S. Lee and J. A. Ventura. An algorithm for constructing minimal c-broadcast networks. *Networks*, 38(1):6–21, 2001.

[70] X. Lin and L.M. Ni. Multicast communication in multicomputer networks. *Parallel and Distributed Systems, IEEE Transactions on*, 4(10):1105–1117, Oct 1993.

[71] C. D. Morosan. *Studies of Interconnection Networks with Applications in Broadcasting*. PhD thesis, Concordia University, Montreal, Canada, 2007.

[72] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[73] J-H. Park and K-Y. Chwa. Recursive circulant: a new topology for multicomputer networks (extended abstract). *Parallel Architectures, Algorithms and Networks, 1994. (ISPAN), International Symposium on*, pages 73–80, 14-16 Dec 1994.

[74] S. Perennes. Broadcasting and gossiping on de bruijn shuffle exchange and similar networks. Technical Report 93-53, I3S, 1993.

[75] R. Ravi. Rapid rumor ramification: approximating the minimum broadcast time. In *FOCS '94 IEEE Symps. on Foundations of Computer Science*, pages 202–213, 1994.

[76] J.-F. Saclé. Lower bounds for the size in four families of minimum broadcast graphs. *Discrete Math.*, 150(1-3):359–369, 1996.

[77] B. Shao. *On k-broadcasting in graphs*. PhD thesis, Concordia University, Montreal, Canada, 2006.

[78] Z. Shen. An optimal broadcasting schema for multidimensional mesh structures. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 1019–1023, New York, NY, USA, 2003. ACM.

[79] P. J. Slater, E. J. Cockayne, and S. T. Hedetniemi. Information dissemination in trees. *SIAM Journal on Computing*, 10(4):692–701, 1981.

[80] J. Stewart. *Single Variable Calculus: Concepts And Contexts*. Thomson Brooks/Cole, 2003.

[81] I. Wojciechowska. *Broadcasting in grid graphs*. PhD thesis, Morgantown, WV, USA, 1999. Chair-Frances L. Scoy.

[82] E. H-k. Wu and C. Chang. Adaptive multicast routing for satellite-terrestrial network. *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, 3:1440–1444 vol.3, 2001.