DEEP HEURISTIC: A HEURISTIC FOR MESSAGE BROADCASTING IN ARBITRARY NETWORKS

Rakshit Majithiya

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

January 2015

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Rakshit Majithiya**

Entitled: **Deep Heuristic: A Heuristic for Message Broadcasting in Arbitrary Networks**

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

Master of Computer Science

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

Signed by the final examining commitee:

_____ Chair

Dr. T. Eavis

_____ Examiner

Dr. J. W. Atwood

_____ Examiner

Dr. G. Butler

_____ Supervisor

Dr. H. A. Harutyunyan

Approved _____

Chair of Department or Graduate Program Director

_____ 20 _____ _____

Amir Asif, Ph.D., Dean

Faculty of Engineering and Computer Science

Abstract

Deep Heuristic: A Heuristic for Message Broadcasting in Arbitrary Networks

Rakshit Majithiya

With the increasing popularity of interconnection networks, efficient information dissemination has become a popular research area. Broadcasting is one of the information dissemination primitives. Finding the optimal broadcasting scheme for any originator in an arbitrary network has been proved to be an NP-Hard problem. In this thesis, a new heuristic that generates broadcast schemes in arbitrary networks is presented, which has $O(|E| + |V| \log |V|)$ time complexity. Based on computer simulations of this heuristic in some commonly used topologies and network models, and comparing the results with the best existing heuristics, we conclude that the new heuristic show comparable performances while having lower complexity.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Dr. H. Harutyunyan for the continuous support of masters study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor.

I would like to dedicate this thesis to my wife for her constant influence and encouragement. I would also like to thank my parents and my family for their endless love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In order to find the best communication structure for parallel and distributed computing, a lot of work has been done in the study of the properties of interconnection networks. The ability to effectively disseminate the information among the processors is an important feature for an interconnection network. There are four main problems regarding information dissemination: broadcasting, accumulating, multicasting and gossiping. The main focus of this thesis is broadcasting and gossiping.

1.1 Problem Statement

Communication efficiency becomes particular important when the network supports a distributed file or database system. There are two approaches to reduce the delay of information dissemination: one is to reduce the amount of data being transferred, while the other is to minimize the delay of information spreading [47].

In addition, broadcasting is a vital communication problem in multiprocessor computer systems. There are many problems that could not be solved by a single processor in an acceptable amount of time. One solution is to divide the problem into subproblems that can be performed in parallel. A

single processor handles one of these subproblems. The result of certain subproblems must be transferred among these processors for further computing [38]. The performance of the communication often determines the efficiency of the interconnected network. Therefore, quick broadcasting is an important goal in parallel systems.

An interconnected network can be modeled as graph $G = (V, E)$, where V is the set of processors, which are referred as vertices (or nodes) and E is the set of communication links referred to as edges. Two nodes $u \in V$ and $v \in V$ are *adjacent* if there is an edge $e \in E$, such that $e = (u, v)$. The degree of a node is the number of neighbors of this node. The degree of graph G is the maximum *degree* among al nodes in this graph. Δ stands for the degree of a graph. A path p in a graph $G = (V, E)$ is a sequence of nodes of the form $p = v_1, v_2, \ldots, v_n, (n \geq 2)$, in which each node v_i is adjacent to the next node. Obviously, the path p is also a sequence of edges. The length of a path is the number of edges in the path. The length of the shortest path between two nodes is the *distance* between them. The *diameter* of a graph is the maximum of the distances between all pair of vertices in the graph. A graph $G = (V, E)$ is said to be *connected* if there is a path between any two vertices on G. It is natural to assume that the network is represented by a graph.

The study of information dissemination problem occurs when the following problem is raised: "There are n ladies, and each one of them knows an item of scandal that is not known to any of the others. They communicate by telephone, and whenever two ladies 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?"[**?**]. This problem, which has become known as the *Gossip Problem*, or the *Telephone Problem*, has in turn been the source of several research papers that have been concerned with the spread of information among a set of people, whether it be by telephone calls, conference calls, letters or even computer networks.

Broadcasting in a network is the process in which a node in the network sends a message to all other nodes. The main difference between broadcasting and gossiping is that, in gossiping, each

node has a different message that will be sent to all others during the process, while only one node has one message to send to all other nodes in broadcasting.

Communication networks can be classified into three types, depending on where the communication bottleneck occurs [12].

1. If, during communication, a processor can only use one of its links, we call this situation *processor-bound* because processors can not quickly relay messages and will hamper the efficiency of the network. This pattern is also called *1-port* or *whispering*.

2. On the contrary, when a processor can use all of its links at the same time, communications are said to be *link-bound*, because it is now the number of links that limits the communications. This pattern is also called *n-ports* or *shouting*.

3. Between these two extreme possibilities, we have the case where a processor can only use k links at the same time.

In order to formalize the gossip problem, let us assume that each node in a graph has a token that needs to communicate to all of the other nodes in the graph. The tokens can be combined so that all communications involve constant time. The time needed for combining is irrelevant and treated as zero. Thus a formal definition can be stated as follows:

Initialization: Let $G = (V, E)$ be a graph (interconnection network). Each node, v, is associated to an initial singleton set $S(v)$, which is the initial data. These initial singleton sets are disjoint.

Allowable Steps: Each node can send its set to a neighbor or neighbors / or receive a set from a neighbor or neighbors depending on the model of communication used. After receiving some sets, a node unites its existing set with all sets received at that step thus forming a new set for the next step.

Final state: All nodes must have the same set locally, containing all elements in the initial singleton set [7].

The action of exchanging tokens between two nodes is referred to as a call. Then, for gossiping, we have the following constraints in the model considered in this thesis:

- A node can only call one adjacent node per unit of time.
- A node can participate in only one call per unit of time.
- Two-way mode is used, that is, a node sends its set and receives a set from its neighbor at the same time.
- Each call requires one unit of time.
- Many calls can be performed in parallel.

In order to measure the gossip time, we employ the term round. In gossiping, a round is a set of parallel calls in the same time unit. A solution to a gossip problem is a sequence of feasible rounds that finish the communication.

In broadcasting we assume that a source node in a graph has a token that needs to be sent to all the other nodes in the graphs. The formal definition for broadcasting is simply stated:

Initialization: Let $G = (V, E)$ be a graph (interconnection network). A source node v is associated with an initial unique token (the initial data). None of other nodes has a token.

Allowable Steps: Depending on the model of communication used, the nodes that have received this token can send it to a neighbor or neighbors who have not yet received this token.

Final state: All nodes must have this token locally.

For broadcasting, we have the following constraints in the model considered in this thesis:

- A node can only call one adjacent node per unit of time.
- A node can participate in only one call per unit of time.
- Each call requires one unit of time.
- Many calls can be performed in parallel.

For broadcasting, a round is a set of parallel calls in the same time unit. The number of rounds is used to measure the broadcast time. Since one round spends one unit of time, the number of

rounds is equal to the total time-steps needed for broadcasting. Given a graph G, the broadcast time $b(G, u)$, or simply $b(u)$, is the minimum broadcast time of graph G originated at node u.

1.2 NP-Completeness

A problem is in class NP if a given solution to this problem can be verified in polynomial time. A problem is said to be *NP-Complete* if it is *NP* and it is as difficult as any other *NP-complete* problem.

At first glance, since the definition of broadcasting is straightforward, broadcasting problems do not seem very hard. However, as many other apparently simple problems, broadcasting problems were proved to be intractable. To prove that a problem is NP-complete, one must first show that it is NP, and second to show that some known NP-complete problem is reducible to it. In [41], it is presented that the problem of determining $b(u)$ for an arbitrary vertex u in an arbitrary graph G is NP-complete. The problem used as a known NP-complete problem is the three-dimensional matching problem (3DM), which was shown to be NP-complete in [15]. The 3DM problem is reduced to the broadcast problem in polynomial time. Below we present the proof given in [41].

The proof shows that the 3DM problem is reducible in polynomial time to a more general *Broadcast Time* problem in which at round 0 a set of vertices already has the message and wants to broadcast it to the rest of the graph. The particular case when the set of originator vertices contains only one originator vertex represents our broadcast problem of determining $b(u)$ for an arbitrary vertex u in an arbitrary graph G.

The general broadcast time problem is formally defined as follows. Given a graph $G = (V, E)$ with a specified set of vertices $V_0 \subseteq V$ and a positive integer k, is there a sequence $V_0, E_1, V_1, E_2, V_2, \ldots, E_k, V_k$ where $V_i \subseteq V$, $E_i \subseteq E(1 \leq i \leq k)$, $E_i = \{u, v\}, u \in V_{i-1}, v \notin V_{i-1}$, $V_i = V_{i-1} \cup \{v\}$ and $V_k = V$. Here k is the total broadcast time, V_i is the set of informed vertices at round i, and E_i is the set of

edges used at round i. It is obvious that when $\mid V_0 \mid = 1$ then this problem becomes our broadcast problem of determining $b(u)$ for an arbitrary vertex u in an arbitrary graph G.

The 3DM problem is defined as follows: Given sets $X = \{x_1, x_2, \ldots, x_m\}$, $Y = \{y_1, y_2, \ldots, y_m\}$, $Z = \{z_1, z_2, \ldots, z_m\}$ and $M \subseteq X \times Y \times Z$, there exists a subset $N \subseteq M$ and $\mid N \mid = m$ such that every two elements in N disagree in all three coordinates [41].

Starting from the sets X, Y and Z in the 3DM problem, a graph G is constructed in polynomial time m as shown in Figure 1, adapted from [41]. First, each vertex $(x_i, y_j, z_k) \in M$ is connected to vertices x_i of X, y_i of Y and z_k of Z. For example, vertex (x_1, y_2, z_3) is connected to x_1, y_2 and z_3. Second, create a set of vertices V_0 containing a vertex for each vertex in M and construct a complete bipartite subgraph from the independent set V_0 and M. Finally, construct remaining vertices and edges exactly as shown in Figure 1. The proof below shows that the 3DM problem is reducible to a broadcast time problem with $k = 4$ in the graph G.

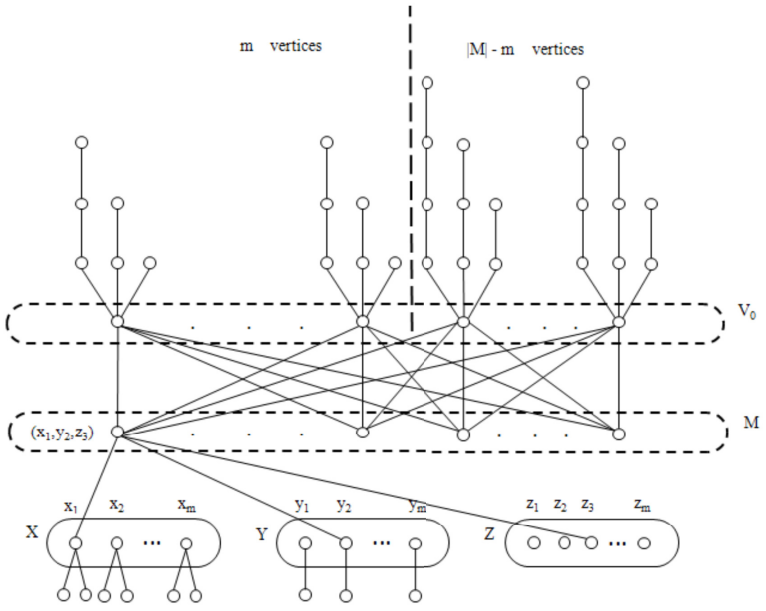


Figure 1: The Graph G Corresponding to the problem 3DM

Given a solution for the broadcast time problem in G, we will show that this is a solution to the broadcast time problem if and only if it is a solution of the 3DM problem. We start by observing that the right side subset of $|M| - m$ vertices of V_0 must start informing the top right vertices in the first round, so that after 4 rounds all vertices on the top right side are informed. Similarly, the left side subset of m vertices must start informing the top left vertices no later than the second round, meaning they are only free for the first round. In order to inform all the vertices on the bottom line in round 4, the left side m vertices in V_0 must inform an m-subset of vertices at round 1 and the vertices in V_0 must be able to inform distinct elements of X, Y and Z at rounds 2, 3 and 4 respectively. This is possible if and only if V_0 is a solution of the 3DM problem.

The next step is to show that 3DM is reducible to determining the broadcast time for an arbitrary graph G with an arbitrary originator u. First construct the graph H shown in Figure 2, as follows. Starting from graph G, add a vertex u, an independent vertex set $U = \{u_1, u_2, \ldots, u_m\}$, and the edges $\{(u, u_i), 1 \leq i \leq m =\mid V_0 \mid\}$. Every vertex u_i joins $m - i$ paths of lengths $6, 7, \ldots, m + 5 - i$. Finally, create a matching between U and V_0 by adding m edges.

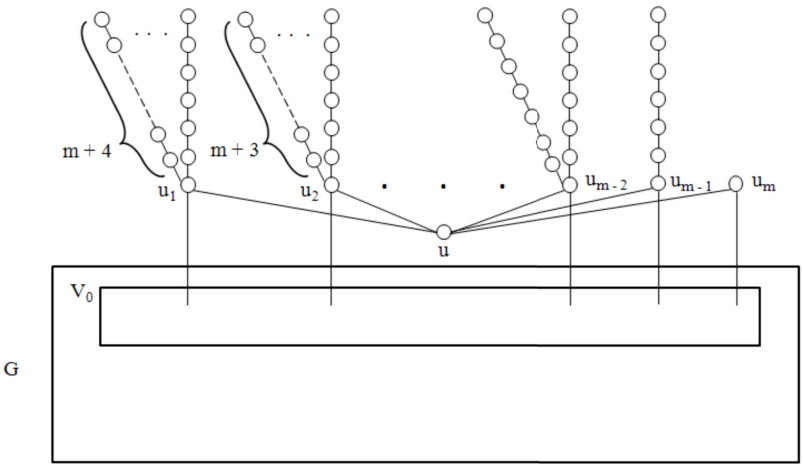


Figure 2: The Graph H

Given the problem to determine whether $b(u) = m+5$ in graph H, consider the following solution. Vertex u will inform each vertex u_i at round i. In turn, each u_i will broadcast the message to the paths connected to it in decreasing path length order. In the end, at time unit $m+1$, u_i informs its

matched vertex in V_0, so that every vertex in V_0 will be informed at round $m+1$. Thus determining if $b(u) = m + 5$ in graph H becomes equivalent to determining if the broadcasting in graph G can be done in 4 rounds, which is the broadcast time problem with $k = 4$ in graph G.

In such cases where a general problem is *NP-Complete*, the research community narrows its focus on more specific instances of the problem. However, the broadcast time problem was proved to also be NP-Complete for specific topologies, such as planar graphs, and bounded degree graphs. In addition, researchers usually also approach the problem with approximation algorithms. Schindelhauer provides results on the inapproximability of the broadcast time problem. In the end, the problem is approached with heuristic algorithms whose results cannot be approximated, but give good simulation results in practice.

Chapter 2

Broadcasting

Literature Review

This chapter reviews some commonly used topologies and the results on broadcasting in these topologies are introduced. Several previously known heuristics for broadcasting are briefly introduced in the second section of this chapter.

It is well known that, for any graph on n nodes, $\lceil \log(n) \rceil \leq b(G) \leq n-1$. Because any informed node can send a message to only one of its adjacent vertex in one time unit, the number of informed nodes can at most be doubled in each round. Thus, at least $\lceil \log n \rceil$ rounds are needed for broadcasting. On the other hand, at least one node must be informed in each round. A situation in which no new node is informed means that the broadcasting has been finished. Therefore, broadcasting takes at most $n-1$ rounds.

For any graph of maximum degree Δ and diameter D, where $D \leq b(G) \leq \delta D$, because it is possible to broadcast in any shortest path spanning tree of G of height D and maximum degree Δ in at most ΔD rounds. In [12], the following lemma is proven. This will be used when discussing broadcast time in several topologies.

Lemma 2.0.1 *In any graph of diameter D, if three different nodes u, v_1, and v_2, with both v_1 and*

v_2 at a distance D from u exists, then $b(G) \geq D + 1$.

2.1 Commonly Used Topologies

Many commonly used topologies and their broadcast times are presented in the three surveys: [12], [23] and [25] This section reviews the commonly used topologies on basis of three important communication parameters: (i) the degree, (ii) the diameter, and (iii) the broadcast time.

The path P_n is a tree with two end nodes of vertex degree 1, and the remaining $n-2$ nodes of vertex degree 2, thus the maximum degree of P_n is 2. The $D(P_n) = b(P_n) = n - 1$. A path is therefore a graph that can be drawn so that all of its vertices and edges lie on a single straight line. Figure 3 shows a path with six vertices, where $D(P_6) = b(P_6) = 5$.

Figure 3: The Path graph for $n = 6$

The Cycle C_n

Cycle C_n, $n \geq 3$, is a simple graph with vertices $v_1, \ldots, v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$. In other words cycle C_n is a path such that the start vertex and end vertex are also connected by an edge. C_n has n vertices and the maximum degree is 2. The $\mathrm{D}(C_n) = \lfloor \frac{n}{2} \rfloor$ and the $\mathrm{b}(C_n) = \lceil \frac{n}{2} \rceil$. Figure 4 demonstrates C_4 and C_6, where the diameter and broadcast time of C_4 is 2 and that of C_6 is 3.

The Complete graph K_n

A complete graph K_n is a simple graph with exactly one edge between any pair of distinct vertices. K_n has n vertices and degree $n - 1$. The diameter of K_n is 1. K_n is a broadcast graph because during each time unit the number of informed vertices is doubled, thus $b(K_n) = \lceil \log_2 n \rceil$. Figure 5

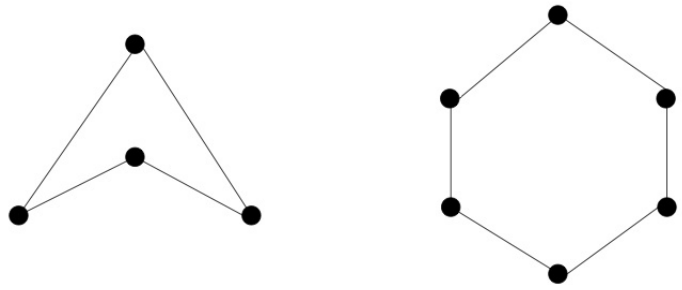


Figure 4: The Cycle graph for $n = 4$ and $n = 6$.

demonstrates K_4 and K_6, where the broadcast time of K_4 is 2 and that of K_6 is 3.

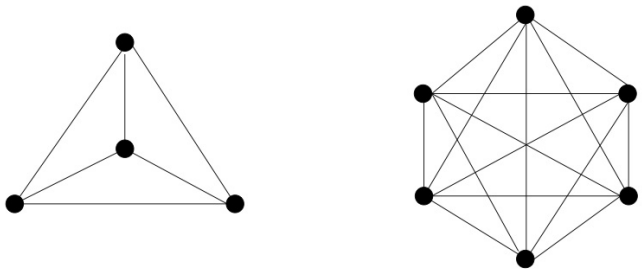


Figure 5: The Complete graph for $n = 4$ and $n = 6$.

The Hypercube H_n

The hypercube of dimension n, denoted by H_n, is a simple graph with vertices representing 2^n bit strings of length $n, n \geq 1$ such that adjacent vertices have bit strings differing in exactly one bit position. H_n has 2^n vertices and $n \cdot 2^n - 1$ edges. The diameter of H_n is n and each vertex has exactly degree n. A $(n+1)$-dimensional hypercube can be constructed from two $n-$dimensional hypercubes by connecting each pair of the corresponding vertices. H_n is the minimum broadcast graph. The $\mathrm{b}(H_n) = \lceil \log_2 2^n \rceil = n$. Figure 6 illustrates three hypercubes of dimensions 1, 2 and 3.

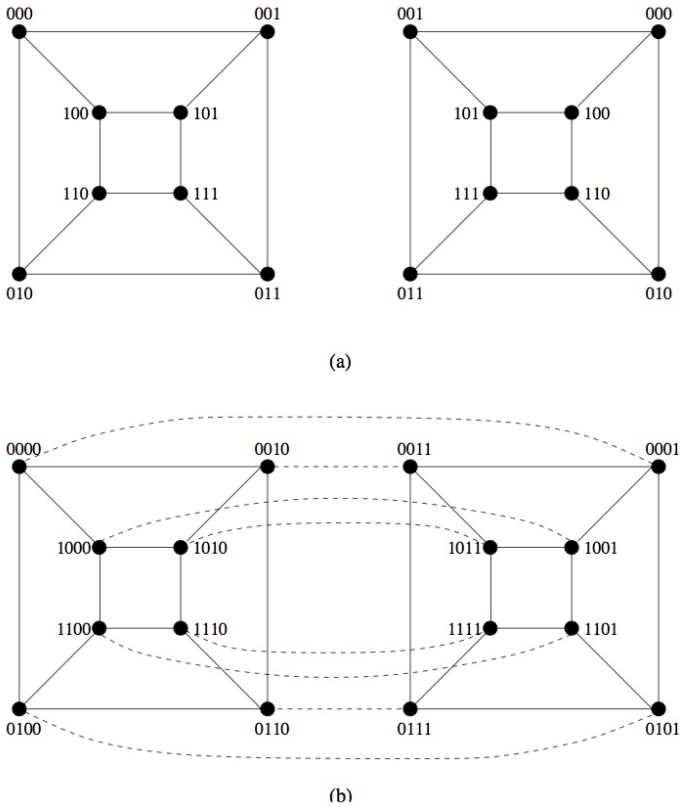


Figure 6: The Hypercube graphs

The Cube-Connected Cycles CCC_n

CCC_n is a modification of the hypercube H_n by replacing each vertex of the hypercube with a cycle of n vertices. The i^{th} dimensional edge incident to a node of the hyper-node is then connected to the i^{th} node of corresponding cycle of the CCC_n. Thus, CCC_n has $n \cdot 2^n$ nodes and the maximum degree is 3. The $\mathrm{D}(CCC_n) = 2n + \lceil \frac{n}{2} \rceil - 2$. The $\mathrm{b}(CCC_n) = \lceil \frac{5n}{2} \rceil - 1$, first every informed vertex sends the message to the hypercube neighbor, then to the right neighbor on the ring, and finally to the left one. Figure 7 shows a 3-dimensional cube connected cycle.

The Shuffle-Exchange SE_n

SE_n is the graph whose vertices can be represented by binary strings of length n. Each edge of SE_n connects vertex βa, where β is a binary string of length $n-1$ and a is in $\{0, 1\}$, with vertex βc and vertex βa, where c is the binary complement of a. SE_n has 2n vertices and the maximum

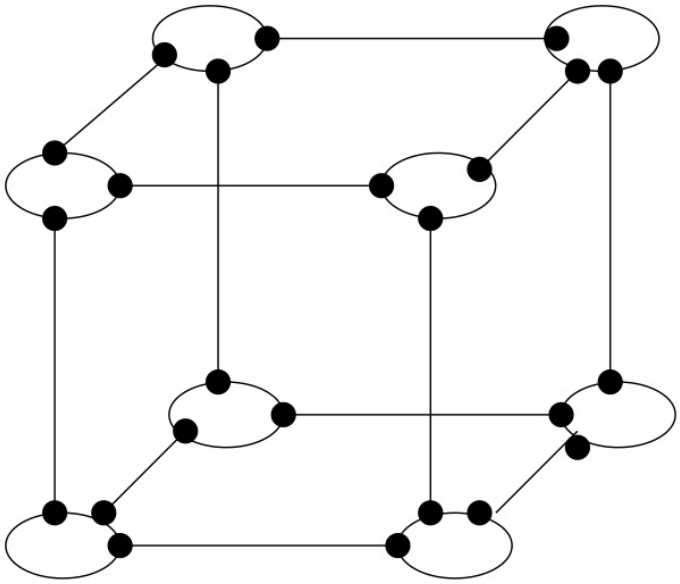


Figure 7: The CCC_3

degree is 3. The D(SE_n)=$2n-1$ and it is provided that $b(SE_n) \leq 2n-1$. Figure 8 represents a Shuffle-Exchange graph SE_3.

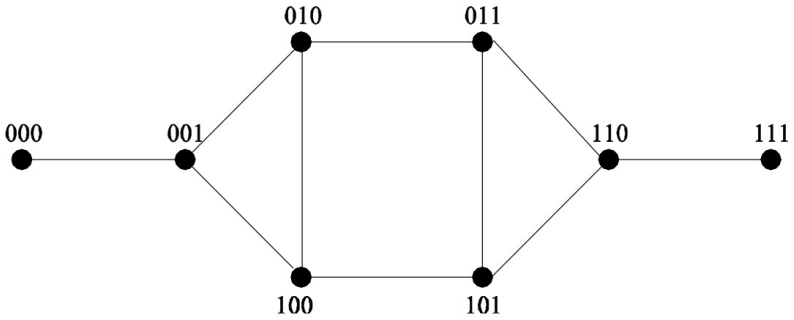


Figure 8: Shuffle Exchange Graph SE_3

The deBruijn DB_n

DB_n is the graph, whose nodes can be represented by binary strings of length n and whose edges connect each string βa, where β is a binary string of length $n1$ and a is in $\{0,1\}$, with the strings βb, where b is a symbol in $\{0,1\}$. DB_n has 2^n vertices with the maximum degree 4 and the diameter is n. [43] provides the lower bound $b(DB_n) \geq 1.3171n$, and [4] proves the upper bound,

$b(DB_n) \leq 1.5n + 1.5$. Figure 9 illustrates a deBruijn graph of dimension 3.

6

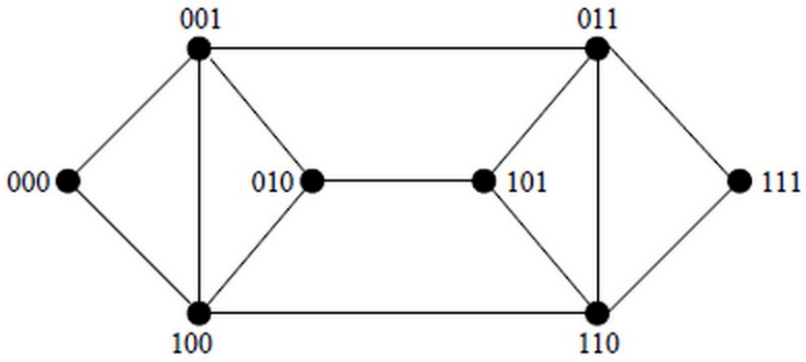


Figure 9: deBruijn Graph DB_3

The d-Grid $G[a_1 \times a_2 \times \cdots \times a_d]$

The d-dimensional grid (or mesh) is the graph whose nodes are all d-tuples of positive integers $(z_1, z_2, \ldots, z_d)$, where $0 \leq z_i < a_i$ for all $i(1 \leq i \leq d)$, and whose edges connect d -tuples, which differ in exactly by one coordinate. For example, in $G[3, 3]$, vertex $(1, 1)$ is connected to vertices $(0, 1), (2, 1), (1, 0)$ and $(1, 2)$. $G[a_1 \times a_2 \times \cdots \times a_d]$ has $a_1 \times a_2 \times \cdots \times a_d$ vertices with the maximum degree $2d$, if each a_i is at least 3. The diameter of d-Grid G$[a_1 \times a_2 \times \cdots \times a_d]$ is $(a_1 - 1) + (a2 - 1) + \cdots + (a_d - 1)$ and [23] provides the $b(G[a_1 \times a_2])$=$a_1 + a_2 - 2$. Figure 10 shows a 2-Grid graph $G[4 \times 5]$.

The d-Torus T

A d-Torus graph is a d-grid graph with both ends of rows and columns connected. $T[a_1 \times a_2 \times \cdots \times a_d]$ denotes the d-Torus graph. The diameter of $k \times k$ X-Torus is given in [11], that is $\lfloor \frac{k}{2} \rfloor + 1$ if k is odd, and $\lfloor \frac{k}{2} \rfloor$ if k is even. It is proven in [11] that the optimal broadcast time of 2-Torus graph is $\lceil \frac{a_1}{2} \rceil + \lceil \frac{a_2}{2} \rceil$, when a_1 or a_2 is even; and it is $\lceil \frac{a_1}{2} \rceil + \lceil \frac{a_2}{2} \rceil - 1$ when both a_1 and a_2 are odd. The

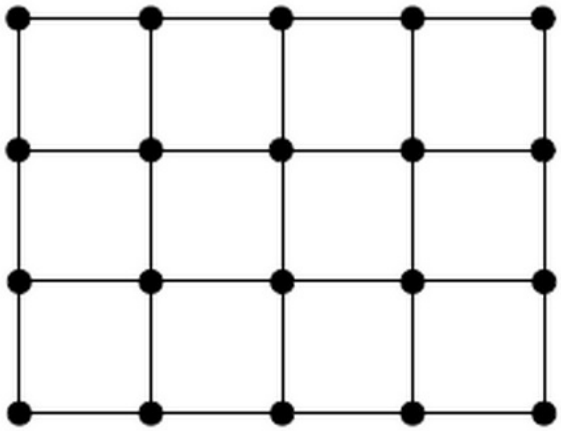


Figure 10: 2- Grid graph $G[4 \times 5]$

bounds on the broadcast time of Torus are $D \leq b(T[a_1 \times a_2 \times \cdots \times a_d]) \leq D + max(0, m-1)$, where $D = \sum_{i=1}^{d} a_i - d$, and m is the number of odd a_i. Figure 11 shows a $2-Torus$ graph $T[4 \times 3]$.

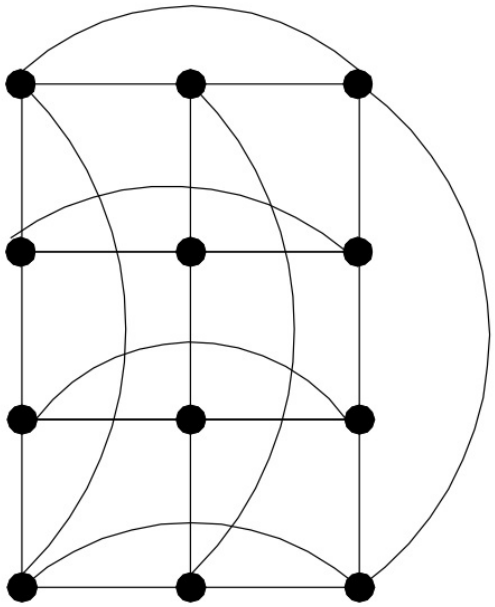


Figure 11: 2-Torus graph $T[4 \times 3]$

Recursive Circulant graph $G(n, d)$

The recursive circulant graph $G(N, d)$ was introduced by Park and Chwa. The recursive circulant graph is defined as $G(N, d) = (V, E)$ with $d \geq 2$, to be a graph where, $V = 0, 1, \ldots, n-1$, and the edge set $E = \{uv | \exists i, 0 \leq i \leq \lceil \log(n) \rceil - 1$, such that $u + d_i \equiv v (\text{mod } n)\}$. $G(N, d)$ has recursive structure when $N = cd^m$, $1 \leq c < d$. The diameter of this graoh is as follows: if d is odd, $D(G(cd^m, d)) = \lfloor \frac{d}{2} \rfloor m + \lfloor \frac{c}{2} \rfloor$. When d is even and c is odd, the diameter is $\lceil \frac{d-1}{2} \rceil m + \lfloor \frac{c}{2} \rfloor$ Finally, when both d and c are even, the diameter is $\lfloor \frac{d-1}{2} \rfloor m + \lfloor \frac{c}{2} \rfloor$. $G(2^m, 4)$, whose degree is m, compares favorably to the hypercube H_m. $G(2^m, 4)$ has the maximum possible connectivity, and its diameter is $\lceil \frac{3m-1}{4} \rceil$. The broadcast time of $G(2^m, 4)$ is m. Figure 12 shows the two recursive circulant graphs, $G(8, 4)$ and $G(16, 4)$.

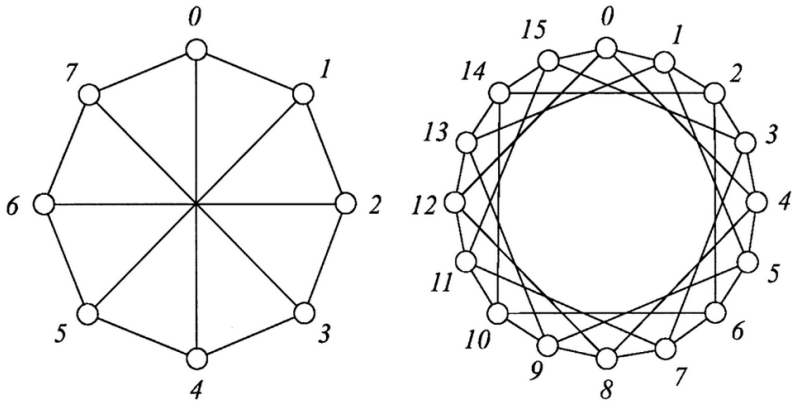


Figure 12: Recursive Circulant graphs $G(8, 4)$ and $G(16, 4)$

Graph	Vertices	Edges	Diameter	Degree	$b(G)$
$P_n(n \geq 3)$	n	$n-1$	n	2	$n-1$
$C_n(n \geq 3)$	n	n	$\lfloor \frac{n}{2} \rfloor$	2	$\lceil \frac{n}{2} \rceil$
K_n	n	$\frac{n(n-1)}{2}$	1	$n-1$	$\lceil \log_2 n \rceil$
H_d	2^d	$d \cdot 2^{d-1}$	d	d	d
$T(n_1, \ldots, n_d)$	$\prod_{i=1}^{d} n_i$	$d \prod_{i=1}^{d} n_i$	$\sum_{i=1}^{d} \lfloor \frac{n_i}{2} \rfloor$	$2d$	$\sum_{i=1}^{d} \lfloor \frac{n_i}{2} \rfloor \leq D \leq \sum_{i=1}^{d} \lfloor \frac{n_i}{2} \rfloor + max\{0, \sum_{i=1}^{d} n_i \bmod 2 \text{ -1 }\}$ [12]
$G(n_1, \ldots, n_d)$	$\prod_{i=1}^{d} n_i$	$\prod_{i=1}^{d} (n_i - 1)$	$\sum_{i=1}^{d} n_i - d$	$2d$	$\sum_{i=1}^{d} n_i - d$ [11]
CCC_n	$n \cdot 2^n$	$3n \cdot 2^{n-1}$	$\lfloor \frac{5n}{2} \rfloor - 2$	3	$\lfloor \frac{5n}{2} \rfloor - 2$ [34]
DB_n	2^n	2^{n+1}	n	4	$1.3171n \leq D(DB_n) \leq \frac{3}{2}(n+1)$
SE_n	2^n	$3 \cdot 2^{n-1}$	$2n-1$	3	$2n-1$
$RC(2^m, 4)$	2^m	$m \cdot 2^{m-1}$	$\lceil \frac{3m-1}{4} \rceil$	m	m

Table 1: Properties of the commonly used topologies

2.2 Previously-Known Heuristics

The first algorithm that attempts to solve the minimum broadcast time problem was presented in [46] in 1981, and then another exact algorithm, based on dynamic programming, was designed by Scheuermann and Wu [47] in 1984. A backtracking algorithm for bounded degree networks is described in [19].

Since finding the minimum broadcast time of any originator in an arbitrary graph is NP- complete, many approximation algorithms and heuristics have been presented to determine the broadcast scheme with minimum time cost.

Given a graph $G = (V, E)$ and the originator u, the heuristic in [29] returns a broadcast scheme whose performance is at most $b(u, G) + Diam(u) + 3\sqrt{|V|}$ rounds, where $Diam(u)$ is the diameter of u, and $b(u, G)$ is the optimal broadcast time. Another well-known algorithm, presented in [44], is based on calculating the poise of a graph. The poise of a tree T is defined as the sum of the maximum degree of any vertex in the tree and the diameter of the tree. The poise of a graph G, denoted by $P(G)$, is defined as the minimum poise of any of its spanning trees. Computing the poise of an undirected graph is NP-hard. However, [44] present an $O(nm \log n)$-complexity heuristic to compute a spanning tree of a graph on n vertices and m edges, such that the poise of the tree is within $O(\log n) \cdot P(G) + O(\log^2 n)$. [44] also proves that $b(G) = O(P(G)\frac{\log n}{\log \log n})$. The time complexity of the algorithm is $O(nm \log^2 n)$, and the upper bound of the broadcast time is $O(\frac{\log^2 n}{\log \log n} b(G))$. Theoretically, the best upper bound is obtained by the algorithm presented in [11], which generates a broadcast scheme with $O(\frac{\log |V|}{\log \log |V|} b(G))$ rounds.

Aside from the algorithms that provide good bounds, some algorithms take advantage of other methods to solve the minimum broadcast time problem. A genetic algorithm is presented in [22], which utilizes a global precedence vector to generate a heuristic of complexity $O(mn^3)$. [2] introduced an integer programming formulation that derives a $O(\log n)$ approximation algorithm. [3] provided a general approach for structured communications, which can be applied to solve the minimum broadcast time problem.

In the following sections, two heuristics of value in practice are introduced in detail. The Round Heuristic [3] and the Tree Based Algorithm [21] are both outstanding heuristics, which have almost the same performance in most of commonly used topologies, and generate better performance in three network models from ns-2 simulator [1, 2, 9, 44]. In many variant topologies, their performances are very close to the optimal value or to the lower bound. Thus, they will be used to scale the new heuristics in this thesis. In the next two subsections, these two heuristics will be introduced.

2.2.1 Round Heuristic

The Round Heuristic is described in [3], which also presents the simulation results in several commonly used graphs. From its simulation results, we can say that its performance is quite close or equal to the optimal value. The Round Heuristic is designed for both broadcasting and gossiping problems, and its broadcasting performance will be considered in this thesis.

During each round of broadcasting, every edge in the network will be assigned a weight. Then, a maximum weighted matching will be performed in the network, in order to activate the matched edges. The activated edges will be selected to pass the message during that round. This procedure will continue until the whole network is informed. This procedure will be performed in every round, and that is why this heuristic is called Round Heuristic.

Setting the weights rationally and effectively are the most significant steps. In [3], two different approaches are introduced to set the weight. One is called the *PotentialApproach*, and the other is the *Breadth-First-Search(BFS)* approach. The potential approach assigns each edge (v, w) a weight equal to its poential, defined as the number of messages known by either v or w, but not by both of them. In broadcasting, the weight could only be 0 or 1.

Obviously, the potential approach is simple and requires little storage and runs very fast. However, as a pure local greedy algorithm, it lacks a global view. The BFS approach works much better in this aspect, although its cost is far more expensive. Before going to the details of BFS approach, several definitions should be presented. In a connected graph, the dispersion region $DR(p, t)$ of a

message p is the set of vertices that know p at the beginning of round t. For any vertex $v, distv(p,t)$ denotes the shortest distance in the graph from v to a vertex $w \in DR(p,t)$. The set of border-crossing edges, denoted by $bce(p,t)$, is defined as $bce(p,t) = \{(v,w) \in E | v \in DR(p,t) \text{ and } w \notin DR(p,t)\}$. For any vertex $v \notin DR(p,t), bcev(p,t)$ consists of all edges in $bce(p,t)$ that lie on the shortest path from $DR(p,t)$ to v. Figure 13 illustrates the dispersion region $DR(p,t)$ for a message p. The border-crossing edges, $bce(p,t)$, are drawn in bold. $distv(p,t) = 3$ and $bcev(p,t) = \{e1, e2\}$.

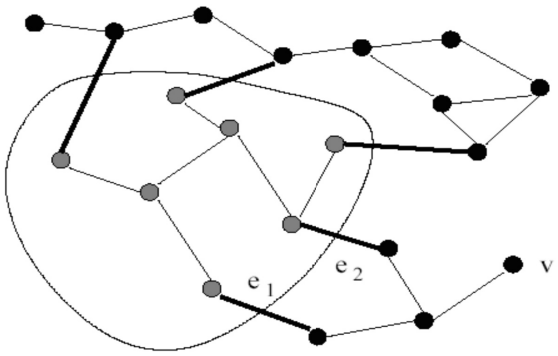


Figure 13: The dispersion region $DR(p,t)$ for some message p.

The weight of an edge is regarded as the sum of the contributions by each message p. Only border-crossing edges can disseminate p further in that round and will be assigned weight. Given an edge $e \in bce(p,t)$, how useful is e for the rapid dissemination of p? Message p should preferably be routed on shortest paths from $DR(p,t)$ to all other vertices: if, for a vertex v, an edge $e \in bcev(p,t)$ is chosen to be active in round t, then $dist_v(p,t+1) = dist_v(p,t)1$. If e lies on many of these shortest paths, it is more useful. The larger $dist_v(p,t)$ is, the more priority should be given to forwarding p towards v. Considering all these criteria, the weight, attributed by all vertices $v \notin DR(p,t)$ to every edge $e \in bcev(p,t)$, is calculated as follows:

$$weight(v,p,t) = \frac{dist_v(p,t)^{Dist\_Exp}}{|bce_v(p,t)|^{Num\_Exp}},$$

where $dist_v(p,t)$ and $bce_v(p,t)$ are calculated for every vertex v, at round t, and $Dist\_Exp$ and Num_Exp are two parameters. [3] applies a modified breadth first search algorithm, such that

vertices are considered in order of increasing $dist_v(p,t)$. For vertex v with $dist_v(p,t) = 1$, $bcev(p,t)$ consists of all incident edges that connect v to a vertex in $DR(p,t)$. For larger $dist_v(p,t)$ the algorithm computes the union of the sets $bce_{w_i}(p,t)$, for all vertices w_i adjacent to v with $dist_{w_i}(p,t) = dist_v(p,t) - 1$. The calculation of the $bce_v(p,t)$ can easily be incorporated into the BFS search.

For any vertex v, $bcev(p,t)$ is the union of at most $|V|$ sets with at most $|E|$ elements each. This computation takes $O(|V|\cdot|E|)$ time. The $bce_v(p,t)$ are calculated for every vertex v. Thus, calculating the weights takes $O(|V|^2|E|)$ in total. Without considering the matching step, the running time of Round Heuristic is $O(R \cdot |V|^2|E|)$, where R is the number of rounds of broadcasting.

The value of the weight depends heavily on the choice of the parameters, $Dist\_Exp$ and Num_Exp. Thus, the impact of the parameters plays a significant role in the performance of the Round Heuristic. Particularly, $Dist\_Exp$ is of great significance, which determines the influence of the distance between nodes and dispersion regions. Usually, values ranging from 0.25 to 60 are used. The precise choice for two topologies are mentioned in [3]: for the mesh graphs, $Dist\_Exp = 4$, while for the butterfly graphs, $Dist\_Exp = 2$. In [3], simulation results of the heuristic in commonly used topologies are presented, including the Cube Connected Cycles, the Shuffle Exchange graphs, the Butterfly graphs as well as the deBruijn graphs. Many of these values are close or equal to the optimal broadcast time, some of which will be presented in Chapter 4.

2.2.2 Tree Based Algorithm

The Tree Based Algorithm (TBA) is presented in [21], whose general idea derives from the Round Heuristic. Same as the Round Heuristic, TBA applies maximum weighted matching to determine the message-passing edges between the bright region and the dark region. In each round, TBA performs a *Breadth-First-Search(BFS)* from $bb(t)$ towards all the uninformed vertices, and the parent-child relationship is determined by labeling each uninformed vertex v with $D(v,t)$. Then every uninformed vertex u will be assigned a weight, which is based on the strategy of the optimal broadcasting in trees. Let $w(u,t)$ stand for the weight of vertex u at round t. If u has no children, then $w(u,t) = 0$. Otherwise, $w(u,t) = max\{w(C_i^u,t)+i\}$, where C_i^u is the i^{th} child of u, and without loss of generality, all the children of u are in decreasing order of their weights. After all the vertices in the dark region are assigned weights, TBA finds a maximum weighted matching between $bb(t)$ and their uninformed neighbors. A matching algorithm with time complexity $O(|E|)$ is applied by the heuristic. The matched edges are used to pass the message in that round. The broadcast time is the round number during which all the vertices in the network are informed. Since each round takes time $O(|V| + |E|) = O(|E|)$, the total complexity is $O(R \cdot |E|)$. [21] also provides simulation results for commonly used topologies as well as some network models, which show that in most cases TBA even has better results than the Round Heuristic. TBA has a refined version, which inherits the idea of choosing parameters from the Round Heuristic, to obtain better broadcast time in some topologies. As a result, the weights are allowed to be decimal in the refinement.

2.2.3 The Minimum-Weight Cover Problem

Another problem that we need to describe is called the Minimum-Weight Cover problem (MWC), which is presented in [29].

Let $G(V_1, V_2, A, w)$ be a bipartite graph with bipartition (V_1, V_2), edge set A, and a weight function $w : A \to Z^+$ on the edges, and no isolated vertices. Each vertex $v_1 \in V_1$ is called a server, and each vertex $v_2 \in V_2$ is called a customer. If a control function $F : V_2 \to V_1$, where $F(v_2) = v_1$

implies $(v_1, v_2) \in A$, and we say that v_1 controls v_2. For every server $v \in V_1$, the clients dominated by v are denoted by $D_1(v), \ldots, D_k(v)$, and the edges connecting v with its clients are denoted by $e_i^v = (v, D_i(v))$. Without loss of generality, all the clients dominated by v are in the order such that $w(e_i^v) \geq w(e_{i+1}^v)$ for $1 \leq i \leq k$.

The MWC problem. Given a bipartite graph $G(V_1, V_2, A, w)$, determine a control function $F{:}V_2 \rightarrow V_1$ whose weight $W(F) = max\{max\{i + w(e_i^v)\}\}$ is minimal. The function F is called the minimum control function for G.

A pseudopolynomial algorithm is given to solve the MWC problem. The basic idea is to check whether there exists a positive integer j, where $min_e\{w(e)\} + 1 \leq j \leq min_e\{w(e)\} + |V2|$, and a control function F, such that $W(F) \leq j$. The algorithm constructs a flow graph G_j based on G and j, with the property that G has a control function F with weight $W(F) \leq j$ iff it is possible to push $|V2|$ units of flow from the source to the sink on G_j. The details of algorithm are as follows.

Algorithm 1 *MWC*

1: Set $j_1 = min_e\{w(e)\} + 1$ and $j_2 = max_e\{w(e)\} + |V_2|$.

2: Let $j = \lceil \frac{j_1 + j_2}{2} \rceil$.

3: Construct the flow graph G_j.

4: Calculate the maximum flow on G_j from source to sink.

5: If the maximum flow is equal to $|V_2|$, then set $j_2 = j$, else set $j_1 = j$.

6: If j equals to j_2 and $j_2 \leq j_1 + 1$, then goto next step, else back to step 2.

7: Return the minimum control function F that corresponds to the maximum flow computed on G_{j_1}.

end

The complexity of the MWC algorithm mainly depends on which maximum flow algorithm is employed. Table 2 shows the different maximum flow algorithms and their time complexities. The Dinic's algorithm runs in $O(|E|\sqrt{|V|})$ time in networks with unit capacities.

Methods	Time Complexity
Ford-Fulkerson algorithm	$O(\|f\|\|E\|)$
Edmonds-Karp algorithm	$O(\|V\|\|E\|^2)$
Dinic's algorithm	$O(\|E\|\sqrt{\|V\|})$
General push-relabel algorithm	$O(\|V\|^2\|E\|)$

Table 2: Theoretical complexities for Hypercube H_d

Another crucial part of this algorithm is the construction of the flow graph G_j. Create a source vertex s and a sink vertex t. Assume that w_v is the maximal weight that is less than or equal to $j1$ of an edge incident to $v \in V_1$. Duplicate v into $w_v + 1$ different copies and arrange the copies in an arbitrary order $v_1, \ldots, v_{w_v} + 1$. For v_1, the first copy of v, create a directed edge (s, v_1) with capacity jw_v and a directed edge (v_1, u) with capacity 1, from v_1 to every customer $u \in V_2$ such that $(v, u) \in A$. For v_i the i^{th} copy of $v, i \geq 2$, create a directed edge (s, v_i) with capacity 1 and a directed edge (v_i, u) with capacity 1 to all the customers u such that $(v, u) \in A$ and $w(v, u) \leq wvi+1$. Finally for each customer $u \in V2$ create a directed edge (u, t) with capacity 1. Figure 16 demonstrates an example of G_3.

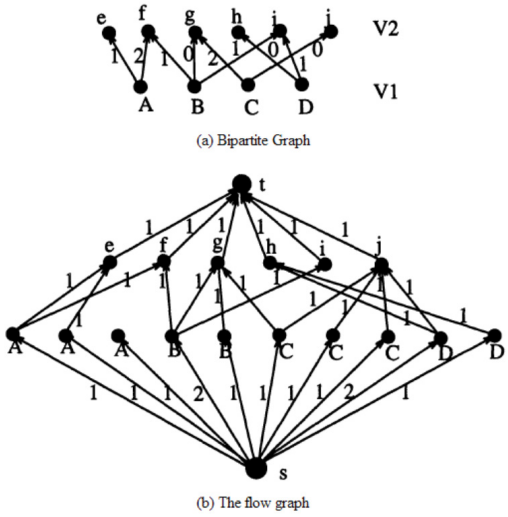


Figure 14: (a) A bipartite graph G. (b) Its corresponding flow graph G_3.

Chapter 3

A New, Improvised Heuristic - Deep Heuristic

3.1 Proposed Heuristic

In this section, the new heuristic is proposed, which has a goal to improve the existing heuristics mentioned previously. Before going in further detail about this proposed heuristic, it would be important to throw some light on different terminologies and concepts used in this heuristic. In the following section, different important definitions are described.

3.1.1 Definitions

Definition 1 *For a given graph G at round t, there are two kinds of regions according to the situation of the message distribution, the Dark Region and the Bright Region. The Dark Region, denoted by $DR(t)$, is a subset of nodes in G that is composed of all uninformed nodes at the beginning of round t. Those nodes in $DR(t)$ that have informed neighbors, compose the dark border, denoted by $db(t)$. The bright border $bb(t)$ is composed of those informed nodes that have uninformed neighbors. The Edges that cross between the Dark Region and the Bright Region are called cross board edges,*

which are denoted by cbe(t).

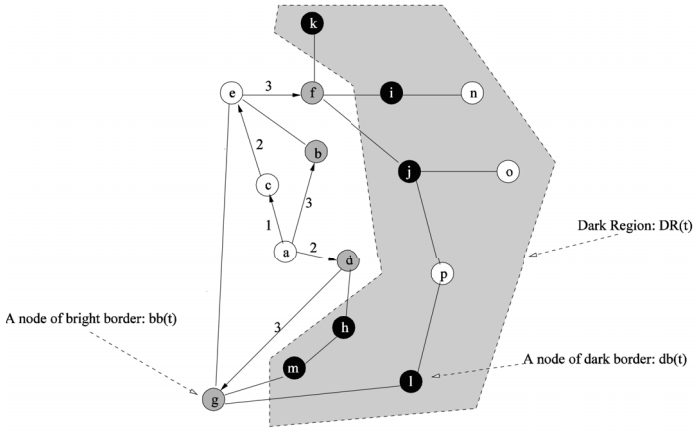


Figure 15: Definitions of Graph Parts

Figure 15 illustrates how these concepts are defined. The dark region $DR(t)$ is represented by the shadowed area. The nodes in $DR(t)$ with the black backgrounds belong to $db(t)$, for example k, i, j, h, m and l. and the nodes not in $DR(t)$ with shadowed backgrounds belong to $bb(t)$. The edges $(f, k), (f, i), (f, j), (d, b)$ and (g, m) belong to $cbe(t)$.

Definition 2 *For a graph and an uninformed node v at round t, there is a shortest distance from node v to a node in $bb(t)$. The shortest distance is denoted as $D(v, t)$.*

The shortest distance can be used to define the child as follows:

Definition 3 *child, parent and descendants: Given an uninformed vertex u and its uninformed neighbor v, if $D(u, t) = D(v, t) + 1$, one can say u is a child of v, and v is the parent of $u \cdot u$, its children and its children's children are all called v's descendants.*

Based on the definition of decedents, one can define the descendent graph as mentioned in the next definition:

Definition 4 *For a graph and an uninformed node v at round t, one can find a descendant graph for v. This descendant graph consists of node v and all its descendants. This is named as the descendant graph of v, which is denoted by $DG(V, E, v)$, or rather $DG(v)$.*

Definition 5 *Estimated time: in order to estimate the broadcast time of $DG(v)$ in round t, we use $EB(v,t)$. EB(v, t) is defined recursively as follows:*

1. *$EB(v,t) = 0$, if node v has no children.*
2. *If v has k children, $c_1, c_2, \ldots, c_k$, and all these k children are listed in order of $EB(c_i,t) \geq EB(c_{i+1},t)$, then $EB(v,t) = max\{EB(c_i,t)+i\}$, for $1 \leq i \leq k$.*

Based on the definition of estimated time, one can construct an algorithm to calculate $EB(v,t)$, given EB of all children of node v.

Algorithm 2 *Algorithm for Calculating $EB(v,t)$.*

1: **procedure** CALCULATE $EB(v,t)$
2: Find $maxEB(ci,t)$, and denote it by MAX.
3: Create k buckets, and number them from 0 to $k-1$.
4: Consider any child c, if $MAX - i \geq EB(c,t) \geq MAX - i - 1$, put c into the ith bucket. Here, only the minimum value and the number of elements are recorded. $SUM(i)$ denotes the number of elements in the first ith buckets and $MIN(i)$ denotes the minimum value in the ith bucket.
5: Get $EB(v,t) = max\{EB(ci,t)+i\}$.
6: **end procedure**

end

The following lemma is derived based on step 5 of Algorithm 2.

Lemma 3.1.1 *$EB(v,t) = max\{SUM(i) + MIN(i)\}$, for $0 \leq i < k$.*

Proof If a node v has k children, $c_1, c_2, \ldots, c_k$ and these children of v are ordered such that $EB(c_i,t) \geq EB(c_{i+1},t)$, then according to definition 5, we have $EB(v,t) = max\{EB(c_i,t)+i\}$, for $1 \leq i \leq k$. $EB(c_i,t)+i$ is order-weight of c_i. Because $MAX - i \geq EB(c,t) \geq MAX - i - 1$

for any child c in the i^{th} bucket, the maximum difference among EB of the children in this bucket is less than 1. Therefore, in the i^{th} bucket, the child with the minimum EB has the maximum order-weight, which is equal to $SUM(i) + MIN(i)$. Thus, $max\{SUM(i) + MIN(i)\}$, for $0 \leq i < k$ is the maximum order-weight of all the children, which is $EB(v, t)$. ∎

3.1.2 Scope of Improvement in Existing Heuristics

Before going further in the algorithm description, it would be easier to understand the need for this heuristic as well as to understand how it improves existing heuristics by going through the situation where improvement can be made in an arbitrary graph if one goes through the broadcast scheme for that graph using the Tree Based Algorithm.

Consider an arbitrary graph G, a vertex $o \in G$ be the *originator*. We would compute the broadcast scheme of this Graph G using Tree Based Algorithm (TBA) [21].

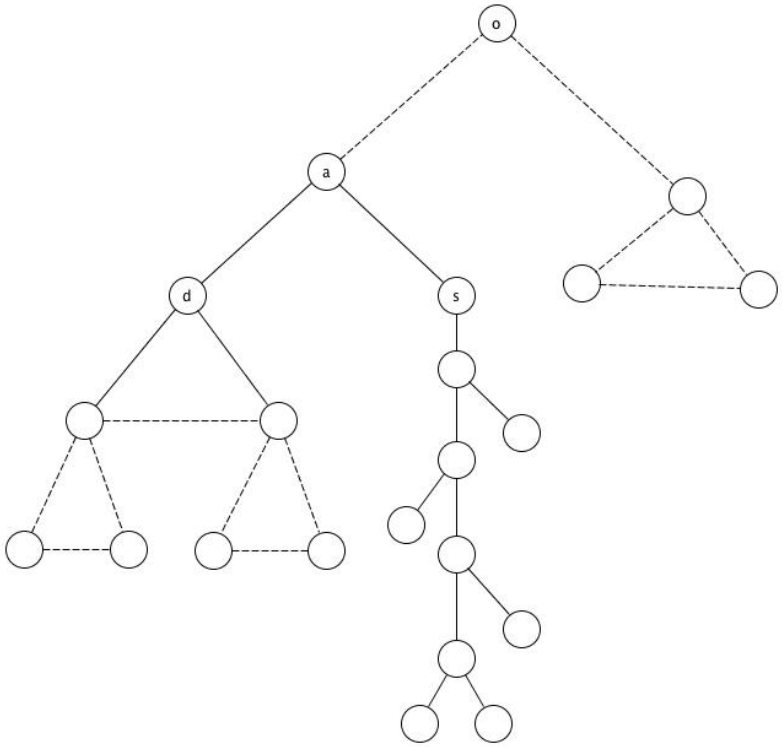


Figure 16: Example Graph G with two subgraphs from vertex a.

If one would perform broadcasting using TBA, as seen in Figure 16 there occurs a situation at round t where the subgraphs originated from vertex d and subgraph s holds different properties in terms of density. The subgraph from vertex d would be dense and the subgraph from vertex s would be sparse. In this round, if $EB(d) \geq EB(e)$, then the information will be sent to vertex d in current time unit t.

If a graph has more density, i.e., if a graph is a dense graph, the broadcast time would be less because on every level there will be more nodes in graph to send information to a specific node in the next level because of its connectivity. At one specific time unit, it is possible that multiple edges are available to convey information from the nodes in the same parent level to the descendant children. With that being said, it is also possible that at one time unit, there will be more number of informed nodes staying idle because other nodes on the same level could be conveying the information to the descendant children on the next level.

So, based on that understanding, in round t describe above, if we prevent the situation where the information is sent to a dense subgraph before the sparse subgraph, one may potentially improve the broadcast time of sparse graph by one time unit. One can certainly say that the information would be delivered to sparse subgraph (originated from vertex s) before the dense subgraph (originated from vertex d), if and only if $EB(s) > EB(d)$. Hence, to make this improvement, one needs to decrease the resulting EB of the dense graph.

In a dense graph, there may be some edges that would remain idle during a broadcast process at one time unit. If we find a way to eliminate those edges during calculation of the broadcast scheme, we can potentially decrease EB of the graph. To find such edges, we can examine the descendants of $bb(t)$ and eliminate the edges that would contribute to increase EB but there might be other edges to the same descendent which can help one vertex in $bb(t)$ to send the information. In the following section a formal algorithm is described, which considers the optimization to potentially decrease the EB of a dense graph.

3.1.3 Algorithm of Proposed Heuristic - Deep Heuristic

Following is the algorithm of the proposed heuristic based on the discussion above.

Algorithm 3 *Deep Heuristic*

1: Initialize $bb(t)$ so that $bb(o)$ has only one node: the originator.

2: Put $EB(v,t)$ as the weight to any node v in $DR(t)$.

3: Sort all vertices in $bb(t)$ by their weight.

4: **Let** $c =$ first child of $db(t)$

5: **Let** $P = ParentsWithSameDescendant(bb(t), c)$

6: **while** $size(P) \neq 1$ **do**

7: **if** $size(P) = 2$ **and** $w(p_0) == w(p_1)$ **and** $deg(p_0) \neq deg(p_1)$ **then**

8: Discard edge $e(p_0, c)$

9: **else**

10: Discard edge $e(p_k, c)$ where $k = min(P)$

11: **end if**

12: **end while**

13: Find the $mnw(t)$ between $bb(t)$ and $db(t)$, and during the process, mark all matched nodes as informed.

14: Compute $bb(t+1)$.

15: If $bb(t+1)$ is empty, the process is complete, and t would be the broadcast time. Otherwise, go to 2.

function ParentsWithSameDescendant(G, c) ▷ G: A set of vertices, c : A vertex $\notin G$

Let R be a set of vertices.

for each vertex v in G **do**

if $\exists\ e(v, c)$ **then**

$R = R \cup \{v\}$

end if

end for

return R

end function

end

3.2 Time Complexity

In this section, we will calculate the time complexity of Algorithm 3.

In step 2 of algorithm 3, the process of assignment of weights to individual nodes is divided in two phases. In the first phase, the heuristic performs a Breadth First Search (BFS) of $DR(t)$ from the $bb(t)$ and labels each node by $D(v,t)$. At the same time, following sets are created:

- A set of nodes which do not have any children. This set is denoted by $rb(t)$, which means remote border.
- A mapping between the set of nodes that are parents of same descendants and their dependents. This can be calculated in a straightforward manner using the *ParentsWithSameDescendant* function.

Let E denote the number of edges in the graph G, then this phase can be calculated in $O(E)$ time. In the second phase, a recursive process is used to compute the weight of each node in $DR(t)$. This process starts from $rb(t)$ towards $db(t)$. In the worst case, one has to calculate $EB(v,t)$ for every single node of the graph G. The degree of the i^th node of graph G is denoted by d_i. By using the deep heuristic to calculate $EB(v,t)$, the time needed for a node with degree d is $O(d)$. Then, the time needed to calculate all the nodes in $\sum_{i=1}^{n} O(d_i)$. Since $\sum_{i=1}^{n} O(d_i) = 2|E|$, the complexity of this phase is $O(E)$. Hence, the total complexity of the weight assignment process in step 2 of algorithm 3 is $O(E)$.

In the 3, we sort all the nodes in $bb(t)$ based on their weight. A node cannot re-appear in bright border at different time units. So, in the worst case, the maximum number of vertices to sort in

step 3 would be the total number of vertices in the graph. Hence, the complexity of step 3 would be $|V| \log |V|$.

Step 5 of the algorithm would be a simple constant time lookup from the mapping of descendent nodes with same parents. We saved this mapping earlier while performing step 2. Hence, it will be a constant time operation. The loop from step 6 to 12 would run for the number of nodes returned in set P. If the graph contains n vertices, then in the worst case, the number of nodes in the set P would be $n - 1$ hence, the complexity of this step would be $|V|$. When using *sort-matching* algorithm in step 13 for matching procedure, the time complexity in one round would be $O(V^2)+O(E) = O(V^2)$. However, when using the *lists-matching* algorithm in the step 13, the time complexity in one round would be $O(E) + O(E) = O(E)$. During the process of matching in round t, whenever a node is marked as informed, it is added to $bb(t)$. If the number of uninformed neighbors of a node in $bb(t+1)$ is 0, then this node will be removed from $bb(t + 1)$. Hence, every single node is present once in the dark border and after getting informed, once in the bright border, making the total time complexity of this operation to be $2 \cdot O(|E|) = O(|E|)$ in the worst case. So, combining all the time complexities of different stages, we get total time complexity of algorithm 3 to be $O(|E| + |V| \log |V|)$.

Chapter 4

Practical Results Using Simulation

This chapter focuses on the evaluation of the Deep Heuristic in practice, presenting its results when run on commonly used network topologies, and on other network topologies, the GT-ITM topology, the Tiers topology, and the BRITE topology from the NS-2 simulator, the most popular network simulator in the network research community.

The results we obtained are compared with the results of all the heuristics presented in the previous chapters.

- The result of Round Heuristic from [3] (RH)
- The Tree Based Algorithm obtained from [21] (TBA)
- The Random algorithm from [51] (P-R)
- The Semi-Random algorithm from [51] (S-R)
- The Minimum-Weight Cover heuristic from [51] (MWC)
- The Minimum-Weight Cover Modified heuristic [51] (MWC-M)

The results are presented in table format with each algorithm on its individual column. In addition to the heuristic abbreviations above, the following abbreviations are also used:

- OPT: The optimal broadcast time in the respective topology
- LOW: The best known theoretical lower bound on the broadcast time in the respective topology
- UP: The best known theoretical upper bound on the broadcast time in the respective topology
- D: The dimension of the topology

In statistics, a confidence interval (CI) is a kind of interval used to indicate the reliability of an estimate of a population parameter. Instead of estimating the parameter by a single value, an interval likely to include the parameter is given. How likely the interval is to contain the parameter is determined by the confidence level or confidence coefficient. Increasing the desired confidence level will widen the confidence interval. A confidence interval is always qualified by a particular confidence level, usually expressed as a percentage; thus one speaks of a "95% confidence interval". The end points of the confidence interval are referred to as confidence limits.

The simulation of the Deep Heuristic is performed 20 times for each graph. Since all samples were of the same value, there was no need to compute the confidence intervals, which is the first advantage of the Deep Heuristic over the existing Random and Semi-Random algorithms.

4.1 Commonly Used Topologies

The commonly used topologies studied in this section are Hypercube (H_d), Cube Connected Cycles (CCC_d), Shuffle-Exchange (SE_d), deBruijn (DB_d) and Butterfly (BF_d). We have already explored the details about these graphs in chapter 2. The results described in following subsections are derived from respective practical implementations of the different heuristics.

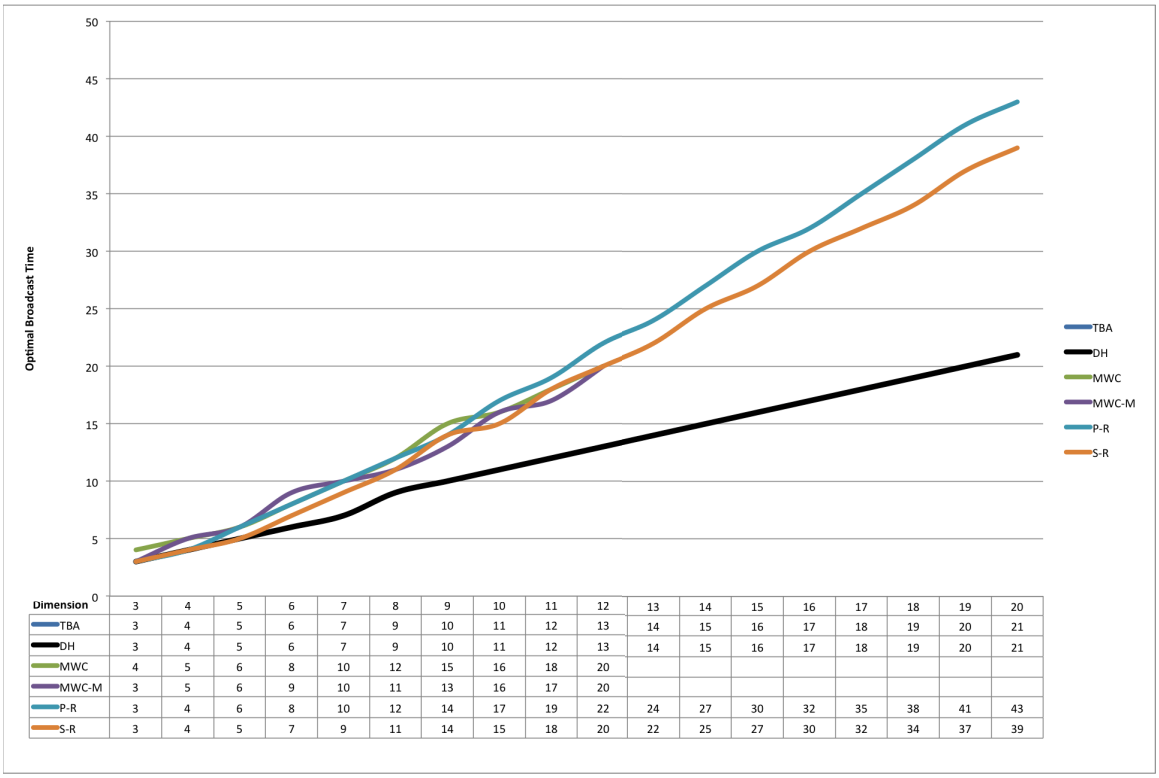
4.1.1 Hypercube (H_d)

We have already seen previously in chapter 2 that the broadcast time of the Hypercube of dimension D is exactly equal to D.The optimal broadcast times of the Hypercube from [24] together with the simulation results of the previous mentioned heuristics and Deep Heuristic are presented in Table 3.

D	**OPT**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
3	3	3	4	3	3	3	3
4	4	4	5	5	4	4	4
5	5	5	6	6	6	5	5
6	6	6	8	9	8	7	6
7	7	7	10	10	10	9	7
8	8	9	12	11	12	11	9
9	9	10	15	13	14	14	10
10	10	11	16	16	17	15	11
11	11	12	18	17	19	18	12
12	12	13	20	20	22	20	13
13	13	14	-	-	24	22	14
14	14	15	-	-	27	25	15
15	15	16	-	-	30	27	16
16	16	17	-	-	32	30	17
17	17	18	-	-	35	32	18
18	18	19	-	-	38	34	19
19	19	20	-	-	41	37	20
20	20	21	-	-	43	39	21

Table 3: Practical results for Hypercube H_d

Figure 17 shows a chart of the simulation results in Hypercubes and we can immediately observe



Dimension	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TBA	3	4	5	6	7	9	10	11	12	13	14	15	16	17	18	19	20	21
DH	3	4	5	6	7	9	10	11	12	13	14	15	16	17	18	19	20	21
MWC	4	5	6	8	10	12	15	16	18	20								
MWC-M	3	5	6	9	10	11	13	16	17	20								
P-R	3	4	6	8	10	12	14	17	19	22	24	27	30	32	35	38	41	43
S-R	3	4	5	7	9	11	14	15	18	20	22	25	27	30	32	34	37	39

Figure 17: Chart of simulation results for Hypercube H_d

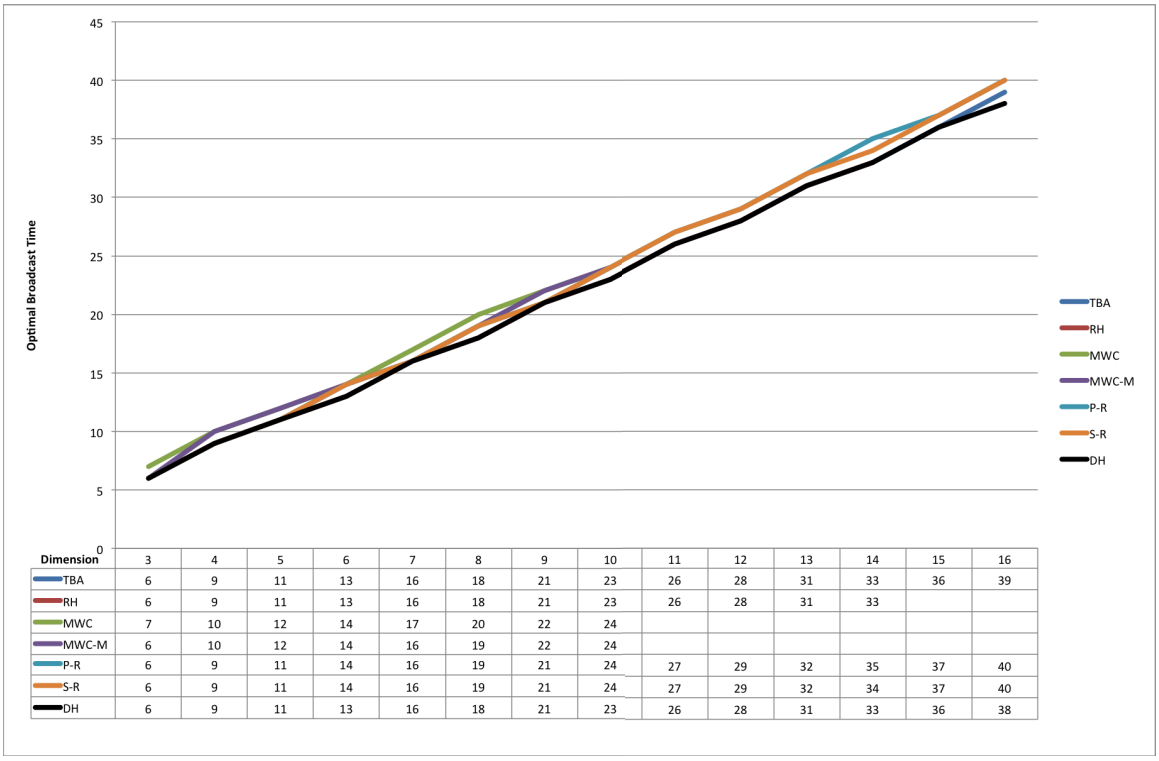
that Deep Heuristic provides optimal broadcast time for all dimensions where simulations were run. It clearly performs much better than all the other previous heuristics. Hence, we can surely say that Deep Heuristic is very suitable for Hypercubes.

4.1.2 Cube Connected Cycles (CCC_d)

The theoretical lower and upper bound in Cube Connected Cycles are presented in [25]. The simulation results of Deep Heuristic are always lower than the theoretical upper bound, usually 1 round less than the upper bound. Compared to the Round Heuristic and the Tree Based Algorithm, which are the previous best heuristics in practice, the Deep Heuristic has similar results. For higher dimensions the results are mostly the same as the best heuristics.

D	**LOW**	**UP**	**RH**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
3	6	7	6	6	7	6	6	6	7
4	9	9	9	9	10	10	9	9	9
5	11	12	11	11	12	12	11	11	12
6	13	14	13	13	14	14	14	14	14
7	16	17	16	16	17	16	16	16	17
8	18	19	18	18	20	19	19	19	18
9	21	22	21	21	22	22	21	21	21
10	23	24	23	23	24	24	24	24	23
11	26	27	26	26	-	-	27	27	26
12	28	29	28	28	-	-	29	29	28
13	31	32	31	31	-	-	32	32	31
14	33	34	33	33	-	-	35	34	33
15	36	37	-	36	-	-	37	37	36
16	38	39	-	39	-	-	40	40	38

Table 4: Practical results for Cube Connected Cycles (CCC_d)



Dimension	3	4	5	6	7	8	9	10	11	12	13	14	15	16
TBA	6	9	11	13	16	18	21	23	26	28	31	33	36	39
RH	6	9	11	13	16	18	21	23	26	28	31	33		
MWC	7	10	12	14	17	20	22	24						
MWC-M	6	10	12	14	16	19	22	24						
P-R	6	9	11	14	16	19	21	24	27	29	32	35	37	40
S-R	6	9	11	14	16	19	21	24	27	29	32	34	37	40
DH	6	9	11	13	16	18	21	23	26	28	31	33	36	38

Figure 18: Chart of simulation results for Cube Connected Cycles (CCC_d)

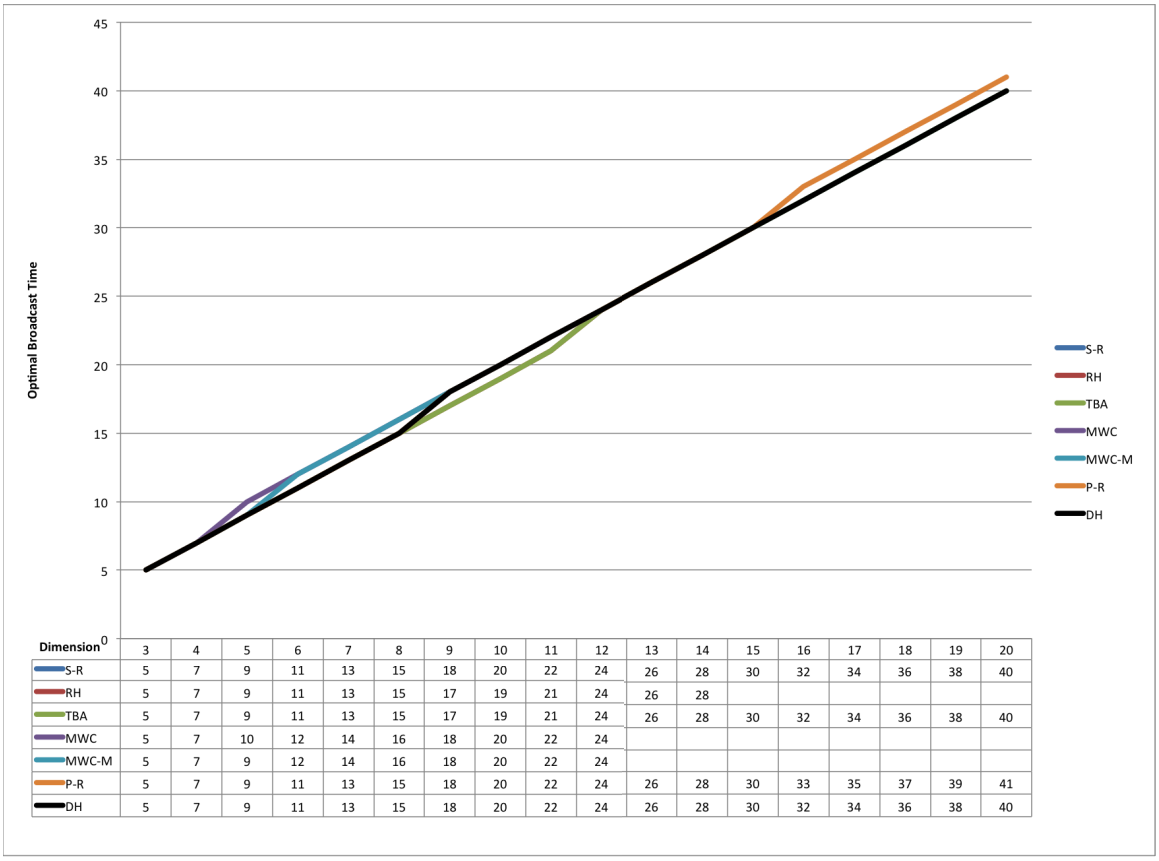
As we can see in the Figure 18, Deep Heuristic starts performing better as the dimension increases. This gives affirmation that for Cube Connected Cycles, Deep Heuristic is very well suitable.

4.1.3 Shuffle-Exchange (SE_d)

The optimal broadcast times in Shuffle-Exchange graphs are presented in [25]. Compared with the previous algorithms with the best performance, the Round Heuristic and the Tree Based Algorithm. When the dimension is less than or equal to 8, the resulting broadcast times are optimal. From dimension 9 and up, the broadcast times are always 1 round more than the optimal , for Deep Heuristic, as well as for the previous best algorithms, the Round Heuristic and the Tree Based Algorithm. Table 5 shows the simulation results in Shuffle-Exchange graphs.

D	**OPT**	**RH**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
3	5	5	5	5	5	5	5	5
4	7	7	7	7	7	7	7	7
5	9	9	9	10	9	9	9	9
6	11	11	11	12	12	11	11	11
7	13	13	13	14	14	13	13	13
8	15	15	15	16	16	15	15	15
9	17	17	17	18	18	18	18	18
10	19	19	19	20	20	20	20	20
11	21	21	21	22	22	22	22	22
12	23	24	24	24	24	24	24	24
13	25	26	26	-	-	26	26	26
14	27	28	28	-	-	28	28	28
15	29	-	30	-	-	30	30	30
16	31	-	32	-	-	33	32	32
17	33	-	34	-	-	35	34	34
18	35	-	36	-	-	37	36	36
19	37	-	38	-	-	39	38	38
20	39	-	40	-	-	41	40	40

Table 5: Practical results for Shuffle-Exchange (SE_d)



Dimension	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
S-R	5	7	9	11	13	15	18	20	22	24	26	28	30	32	34	36	38	40
RH	5	7	9	11	13	15	17	19	21	24	26	28						
TBA	5	7	9	11	13	15	17	19	21	24	26	28	30	32	34	36	38	40
MWC	5	7	10	12	14	16	18	20	22	24								
MWC-M	5	7	9	12	14	16	18	20	22	24								
P-R	5	7	9	11	13	15	18	20	22	24	26	28	30	33	35	37	39	41
DH	5	7	9	11	13	15	18	20	22	24	26	28	30	32	34	36	38	40

Figure 19: Chart of simulation results for Shuffle-Exchange (SE_d)

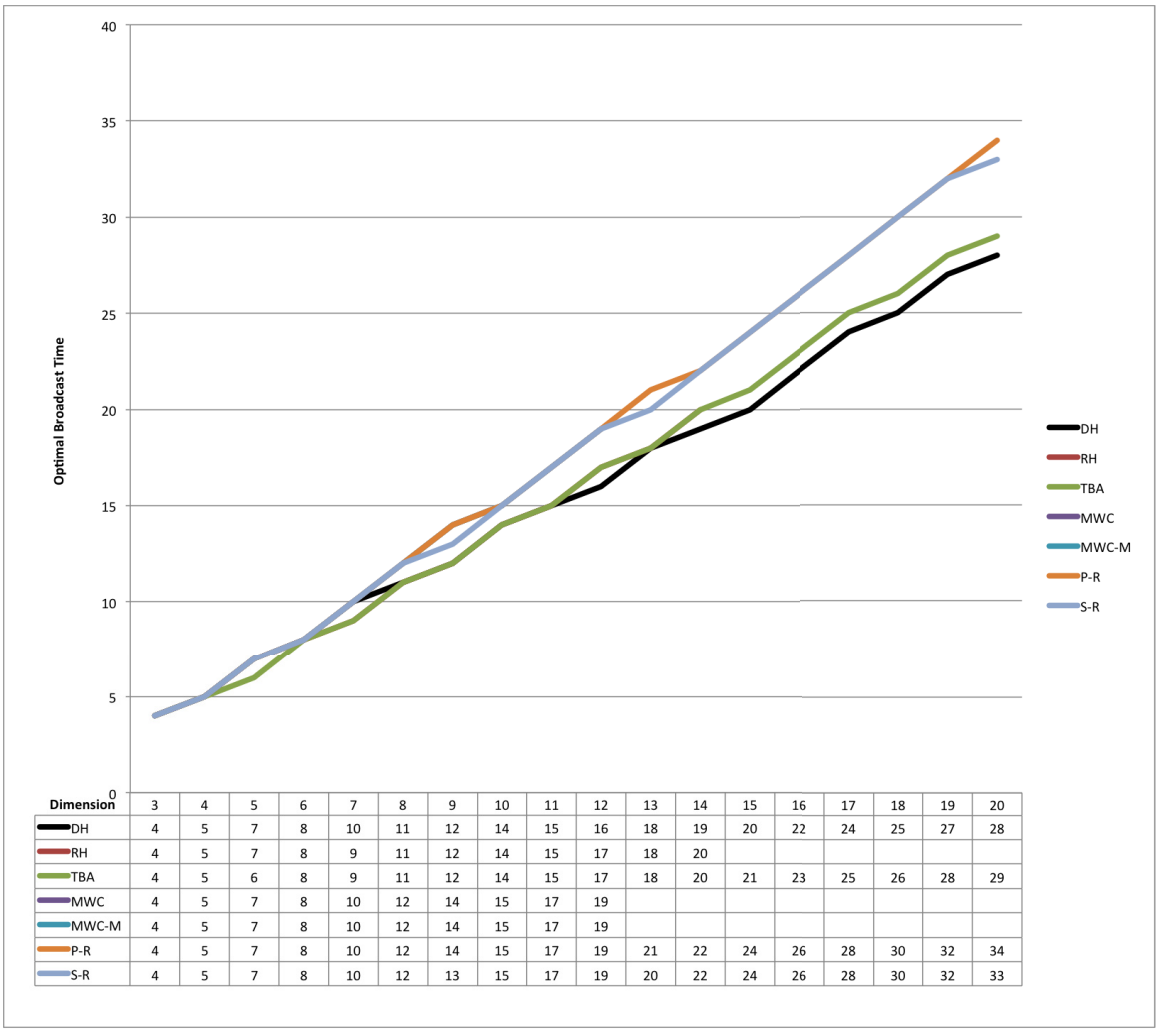
4.1.4 DeBruijn (DB_d)

Table 6 shows the simulation results of different heuristics in DeBruijn graphs as well as the lower and upper bounds of DeBruijn graphs. The lower bounds were calculated using the formulas in [43], and they only hold asymptotically. For this reason, Table 6 shows for dimensions 4 and 5 some broadcast times that are less than the given lower bounds.

D	**LOW**	**UP**	**RH**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
3	4	6	4	4	4	4	4	4	4
4	6	8	5	5	5	5	5	5	5
5	7	9	7	6	7	7	7	7	7
6	8	11	8	8	8	8	8	8	8
7	10	12	9	9	10	10	10	10	10
8	11	14	11	11	12	12	12	12	11
9	12	15	12	12	14	14	14	13	12
10	14	17	14	14	15	15	15	15	14
11	15	18	15	15	17	17	17	17	15
12	16	20	17	17	19	19	19	19	16
13	18	21	18	18	-	-	21	20	18
14	19	23	20	20	-	-	22	22	19
15	20	24	-	21	-	-	24	24	20
16	22	26	-	23	-	-	26	26	22
17	23	27	-	25	-	-	28	28	24
18	24	29	-	26	-	-	30	30	25
19	26	30	-	28	-	-	32	32	27
20	27	32	-	29	-	-	34	33	28

Table 6: Practical results for DeBruijn (DB_d)

The simulation results of Deep Heuristic seems quite similar to the results of TBA. For dimensions less than 13, they are same as TBA, but for dimensions 14 and up, the optimal broadcast time of Deep Heuristic discovered to be less than TBA, so it can be observed that as the dimension increases, Deep Heuristic starts becoming more efficient than other algorithms. Plotting the data represented in Table 6, we can see in Figure 20 how performance of Deep Heuristic changes based on dimension of DeBrujin graph.



Dimension	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
DH	4	5	7	8	10	11	12	14	15	16	18	19	20	22	24	25	27	28
RH	4	5	7	8	9	11	12	14	15	17	18	20						
TBA	4	5	6	8	9	11	12	14	15	17	18	20	21	23	25	26	28	29
MWC	4	5	7	8	10	12	14	15	17	19								
MWC-M	4	5	7	8	10	12	14	15	17	19								
P-R	4	5	7	8	10	12	14	15	17	19	21	22	24	26	28	30	32	34
S-R	4	5	7	8	10	12	13	15	17	19	20	22	24	26	28	30	32	33

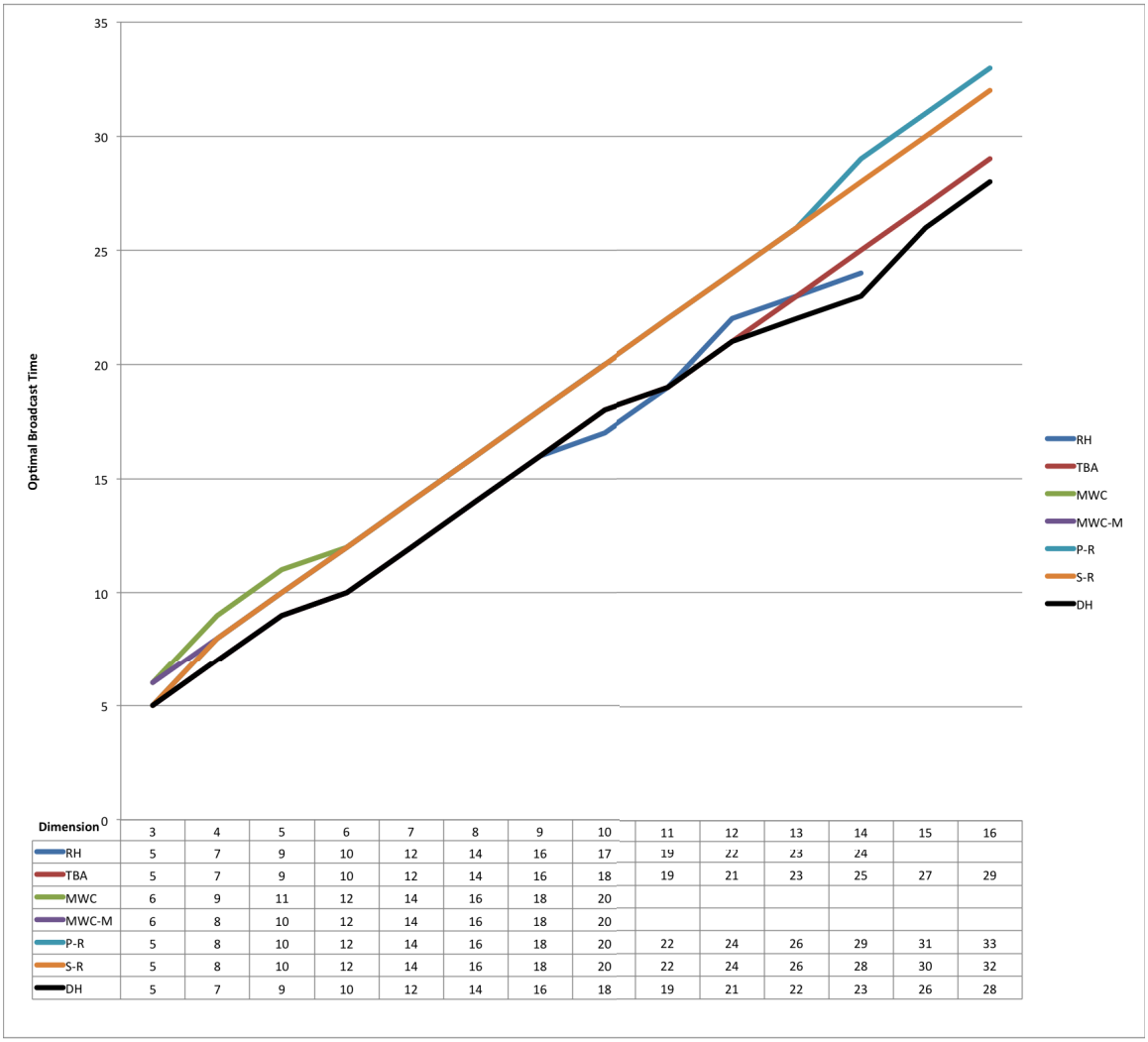
Figure 20: Chart of simulation results for DeBruijn (DB_d)

4.1.5 Butterfly (BF_d)

Table 7 shows the simulation results of the different algorithms in Butterfly graphs. The lower and upper bounds are the ones presented in [25]. The Deep Heuristic performs similar to the Tree Based Algorithm for lower dimension, up to 11. As the dimension increases, the Deep Heuristic is consistently 1 time unit faster than the Tree Based Algorithm. Hence, in this example as well, Deep Heuristic adds value to the algorithm

D	**LOW**	**UP**	**RH**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
3	5	5	5	5	6	6	5	5	5
4	7	7	7	7	9	8	8	8	7
5	8	9	9	9	11	10	10	10	9
6	10	11	10	10	12	12	12	12	10
7	11	13	12	12	14	14	14	14	12
8	13	15	14	14	16	16	16	16	14
9	15	17	16	16	18	18	18	18	16
10	16	19	17	18	20	20	20	20	18
11	18	21	19	19	-	-	22	22	19
12	19	23	22	21	-	-	24	24	21
13	21	25	23	23	-	-	26	26	22
14	23	27	24	25	-	-	29	28	23
15	24	29	-	27	-	-	31	30	26
16	26	31	-	29	-	-	33	32	28

Table 7: Practical results for Butterfly (BF_d) graph



Dimension	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RH	5	7	9	10	12	14	16	17	19	22	23	24		
TBA	5	7	9	10	12	14	16	18	19	21	23	25	27	29
MWC	6	9	11	12	14	16	18	20						
MWC-M	6	8	10	12	14	16	18	20						
P-R	5	8	10	12	14	16	18	20	22	24	26	29	31	33
S-R	5	8	10	12	14	16	18	20	22	24	26	28	30	32
DH	5	7	9	10	12	14	16	18	19	21	22	23	26	28

Figure 21: Chart of simulation results for Butterfly (BF_d) graph

4.2 Simulation Results and Comparisons in NS-2 Models

This section discusses the broadcasting problem in different network models that are popular in the research in interconnection networks community. The simulation results of the Deep Heuristic are presented and comparisons are made with the existing algorithms we previously discussed, Round Heuristic, the Tree Based Heuristic, the MWC and MWC-M heuristics, and the Random and Semi-Random heuristic.

In this thesis we focus on four different network models, GT-ITM Random [9], GT-ITM Transit-Stub [9], Tiers [8], and BRITE Top-down Hierarchical models [37]. These network models have been developed by different research groups and they can all be integrated with the NS-2 simulator. The NS-2 is a simulator used for research in networks and one of its many features is its ability to generate topologies based on different network models.

GT-ITM stands for Georgia Tech Internetwork Topology Models and contains the two models GT-ITM Random and GT-ITM Transit-Stub.

GT-ITM Random model

The GT-ITM Random model uses a pure random generator, which randomly places vertices on a plane and connects each pair of vertices based on a probability . It is obvious that this network model is driven by the probability. Although this random model does not correspond to any real network, it is still presented and discussed in the network research community.

GT-ITM Transit-Stub Model

The GT-ITM Transit-Stub models the Internet. Small networks, such as private company or campus networks called LANs (Local Area Networks) are formed. These are then typically connected together into Metropolitan Area Networks (MANs), which can connect multiple LANs in a larger area, such as city, or Wide Area Networks (WANs), which can be extended to LANs from an entire country or the whole world. The Transit- Stub model regards each independent network as a routing

domain. All the vertices from one independent network are part of the same routing domain and share the same routing information. Routing domains are classified in two types, stub domains and transit domains. Stub domains are local and are concerned with local domain traffic, corresponding to the LANs in the Internet model. Transit domains are global, their goal is to interconnect stub domains and correspond to the MANs or WANs in the Internet model.

Stub domains are usually not connected directly to each other, although it can happen; but typically stub domains are first connected directly to one or multiple transit domains and from thereon indirectly to other stub domains. Depending on whether a stub domain is connected to one or multiple transit domains, the stub domain is called single-homed or respectively multi-homed. A gateway node in the stub domain is connected to a node in the transit domain, which in turn can connect to another node in the same transit domain or in another transit domain or to other gateway nodes from other stub domains. The transit domain nodes are also called backbone nodes.

A method to produce transit-stub graphs by interconnecting transit and stub domains is presented in [54]. This method first generates a connected random graph; each node in that graph represents an entire transit domain. Each node in that graph is then replaced by another connected random graph, representing the backbone topology of one transit domain. Next, for each node in each transit domain, this method generates a number of connected random graphs representing the stub domains attached to that node. Each of these stub domains has an edge to its transit node. Finally, it adds some extra connectivity, in the form of edges between pairs of nodes, one from a transit domain and one from a stub or one from each of two different stub domains. Method parameters control the number of extra edges of each type. Figure 22, adapted from [54], shows an example of such a structure.

The size of the graph (number of nodes) and the distribution of nodes between transit and stub domains in this method are controlled by the following parameters:

- The number of transit domains.
- The average number of nodes per transit domain.

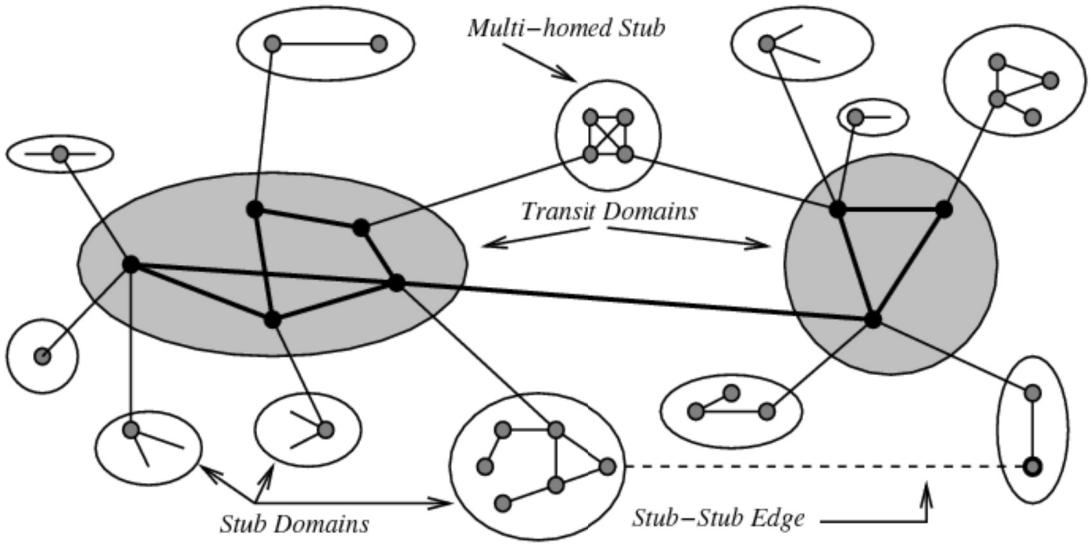


Figure 22: Example of Internet Domain Structure

- The average number of stub domains per transit node.
- The number of average nodes per stub domain.

The following parameters control the total number of edges in the GT-ITM Transit-Stub model:

- The number of transit-stub and stub-stub edges.
- The probability of an edge between each pair of nodes in the transit domains and stub domains.

The Tiers model is one of the most realistic models for generating random networks. Similar to the GT-ITM Transit-Stub, it has the hierarchical domain structure that is present in the Internet. The three levels of hierarchy, the WAN, MAN and LAN levels, are modeled, corresponding to transit domains, stub domains, and LANs attached to stub nodes. The three levels are also called tiers, hence the name Tiers model. The model only supports one WAN.

The Tiers model creates the three hierarchy levels one by one, WAN first, then MANs and finally LANs. The various types of networks are then interconnected according to a given set of parameters. WANs and MANs are created by placing nodes at random in a grid and connecting

them in sub-graphs by joining all the nodes in a single WAN or MAN domain using a minimum spanning tree. Since minimum spanning trees are sometimes used in reality as the basis for laying out large networks, the use of a minimum spanning tree makes the Tiers model more realistic. LANs such as Ethernet and Token Rings are modeled as star topologies. This significantly reduces the number of edges in the graph and reflects the lack of physical redundancy in most LANs. The LAN networks are created by choosing one node in each LAN as the center of the star and connecting every other node to it with a single edge.

The set of parameters below is used to generate a Tiers model network:

- N_W, the number of WANs and S_W, the number of nodes in a WAN. N_W is taken as 1 for simplicity.
- N_M, the number of corporate / institutional networks (MANs) and S_M, the number of nodes per MAN.
- N_L, the number of LANs per MAN and S_L, the number of nodes per LAN.

For $N_W = 1$, the total number of nodes in the graph, N, is given by $N = S_W + N_M \cdot S_M + N_M \cdot N_L \cdot S_L$

The other parameters of the model are:

- The degree of intra-network redundancy in the WAN (R_W), MAN (R_M) and LAN (R_L). This is expressed simply as the degree (number of directed edges) from a node to another node of the same type. So R_L is usually 1, R_M might be 2 and R_W could be 3.
- The degree of internetwork redundancy between networks. This is the number of connections between a MAN and a WAN (R_{MW}) or a LAN and a MAN (R_{LM}).

Figure 23 and Figure 24, adapted from [5], show a typical full internetwork, and respectively a larger internetwork as generated by Tiers. The first one has one WAN with eight nodes, three MANs with three nodes each and two LANs per MAN with three nodes per LAN. The second one

is larger and the endpoints of the links to MAN and LAN nodes have been omitted for clarity, so only the WAN nodes are seen clearly.

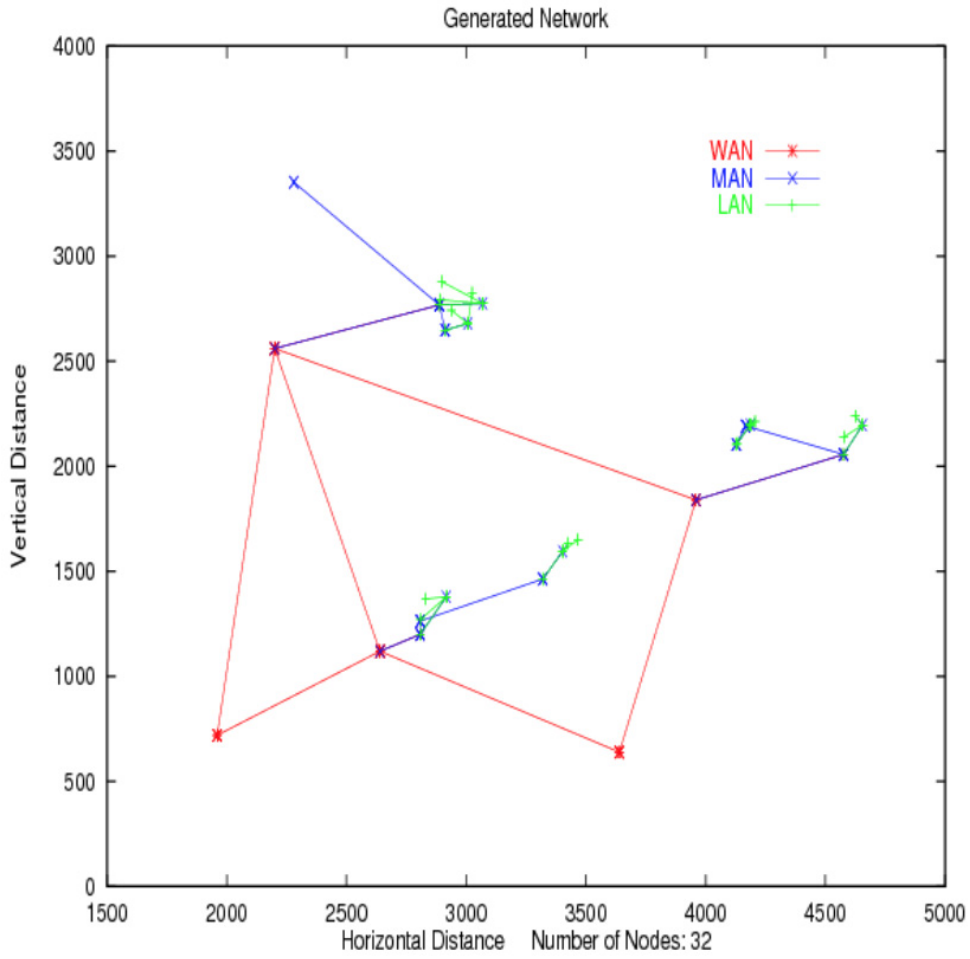


Figure 23: A typical Tiers internetwork

GT-ITM Transit-Stub and Tiers implementations generate networks whose topology resembles typical internetwork. Both implementations are based on the explicitly hierarchical modeling approach described in [5]. Tiers introduces a different method for connecting the nodes in a network, by using a minimum spanning tree, which guarantees connectivity, and produces more realistic networks at the WAN scale. The Transit-Stub implementation uses a smaller set of parameters to control the different aspects of the network, hence takes a more probabilistic approach than that

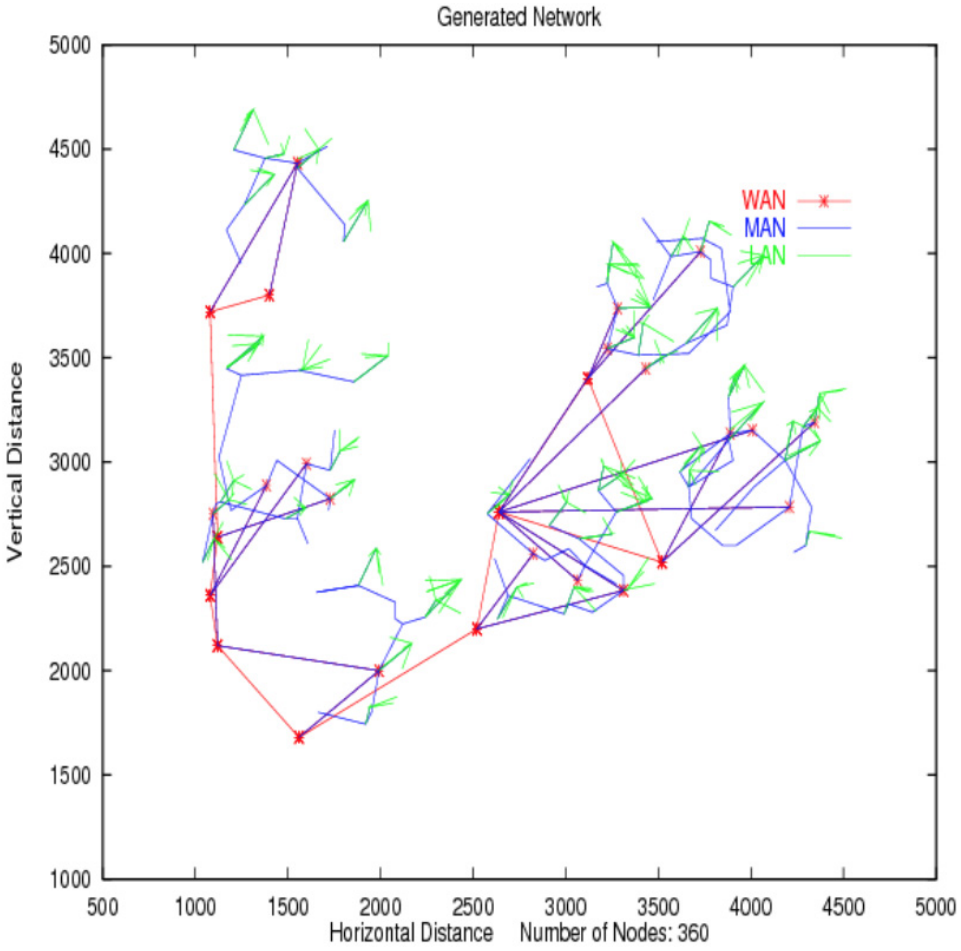


Figure 24: A large Tiers internetwork

of Tiers. In both implementations, most of the parameters can be expected to remain constant between runs of generated networks.

BRITE models

Finally, we briefly present BRITE, the **B**oston university **R**epresentative **I**nternet **T**opology g**E**nerator, which is a topology generation tool that provides a researcher with a wide variety of generation models, as well as the ability to easily extend such a set by combining existing models or adding new ones [37]. BRITE has the capability to work with many different generation models. Some of them are very similar and share implementation code, and others are completely different and share no

functionality. Some can be imported models, such as GT-ITM or Tiers, others can be generated by BRITE, e.g., Flat Router-level Models, Flat AS-level Models, and Top-down Hierarchical Models.

Flat topology models are the early models where the nodes are randomly placed on a Euclidean plane irrespective of any hierarchy order among them as opposed to later hierarchical topology models such as the Tiers and the Transit-Stub. BRITE generates Flat Router-level models in two major steps. First, the nodes are placed on a Euclidean plane randomly or in a heavy-tailed way. When node placement is random, each node is placed in a randomly selected location of the plane. When the placement is heavy-tailed, BRITE divides the plane into squares. Each of these squares is assigned a number of nodes drawn from a heavy-tailed distribution. Once that value is assigned, then that many nodes are placed randomly in the square. Second, edges are added to the graph in one of two ways:

1. Using one of the most commonly used models for generating graphs, Waxman's probability model [52], which considers all possible pairs (u, v) of nodes and uses the probability function P_e, where $P_e(u, v) = \alpha e^{-d/(\beta L)}$ to create an edge, where d is the Euclidean distance between the nodes u and v, L is the maximum possible distance between the two nodes α and β are parameters in the range $0 < \alpha, \beta \leq 1$.

2. Using the Barabasi-Albert (BA) [**?**] model, which connects the nodes according to an incremental growth approach. Incremental growth refers to growing networks that are formed by the continual addition of new nodes, and thus the gradual increase in the size of the network. When a node is added to the network, the probability that it connects to a node already in the network is given by: $P(i, j) = \frac{d_j}{\sum_{k \in V} d_k}$ where d_j is the degree of the target node, V is the set of nodes already in the network and $\sum_{k \in V} d_k$ is the sum of the degrees of all nodes that are already in the network.

Flat AS-level Models represent AS-level topologies. An Autonomous System (AS)-level network is a network under a single administration domain. The AS-level models currently provided by BRITE are very similar to the models provided for generating router-level topologies. The main

difference between these router-level and AS-level models is the fact that AS models place AS nodes in the plane and these can contain associated topologies.

Finally, BRITE also supports generation of hierarchical topologies, currently only of two- level hierarchical topologies. However, two-level hierarchical topologies are in concordance to the two level routing hierarchy that has persisted in the Internet since ARPANET evolved into a network of networks interconnecting multiple autonomous systems.

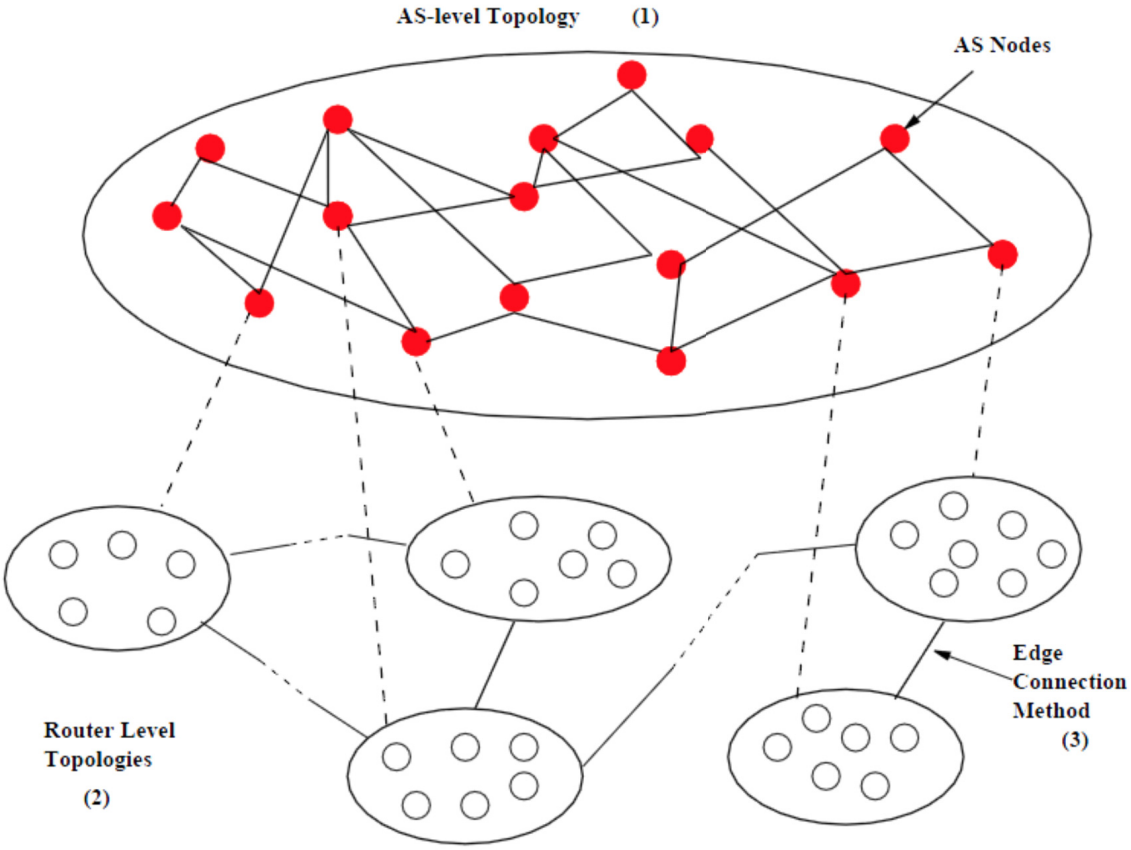


Figure 25: A large Tiers internetwork

BRITE uses a top-down approach to generate hierarchical topologies. Figure 25 adapted from [37], shows a top-down hierarchical model. BRITE first generates an AS-level topology (1) using one of the available flat AS-level models (e.g., Waxman, BA, etc.). Next, for each node in the AS-level topology BRITE will generate a router-level topology (2) using a generation model from the available

flat models that can be used at the router-level. The router-level topologies are interconnected using one of four edge connection mechanisms, borrowed from the popular GT-ITM topology generator. The main goal is to gradually increase the set of edge connection methods with models that reflect what actually happens in Internet topologies.

4.2.1 GT-ITM Random Model

The results of our simulations in the GT-ITM Random model are presented in this chapter. Our results in the GT-ITM graph with 200 vertices are shown in Table 8 and Figure 26, which also show the data collected by the previous algorithms we already discussed. The parameter P is an input parameter to the GT-ITM topology generator and represents the probability of having an edge between each pair of vertices. Obviously, a higher probability leads to more edges in the graph.

We can observe that for graphs with small number of edges (small P), the Deep heuristic performs slightly worse than previous heuristics because of some precalculations in the earlier stages, but as the number of edges increases, the Deep Heuristic tends to perform better than all other heuristics. At one point, Deep heuristic turns out to be the best performing algorithm out of all the other ones and increasing the number of edges further, it can be observed that Deep heuristic performs the same as TBA.

P	Edges	RH	TBA	MWC	MWC-M	P-R	S-R	DH
0.015	316	10	10	11	11	10	10	11
0.016	346	10	10	11	11	10	10	11
0.017	373	10	10	11	11	10	10	11
0.018	388	9	9	11	11	10	10	10
0.019	391	11	11	10	10	10	10	10
0.02	411	9	9	10	10	10	10	9
0.022	423	9	9	10	10	10	10	9
0.024	475	8	8	10	11	10	10	7
0.025	494	9	8	11	11	10	10	8
0.026	507	8	8	11	10	10	10	8

Table 8: Practical results for GT-ITM Random model with 200 vertices

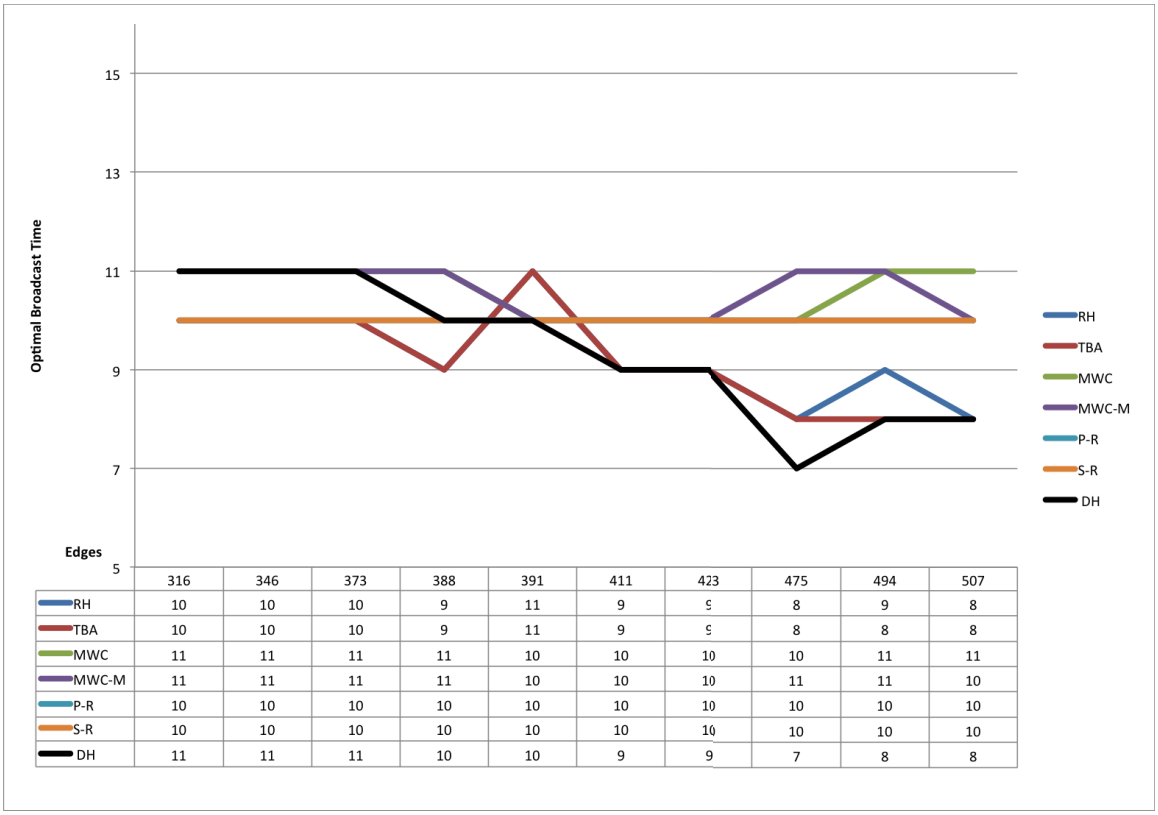
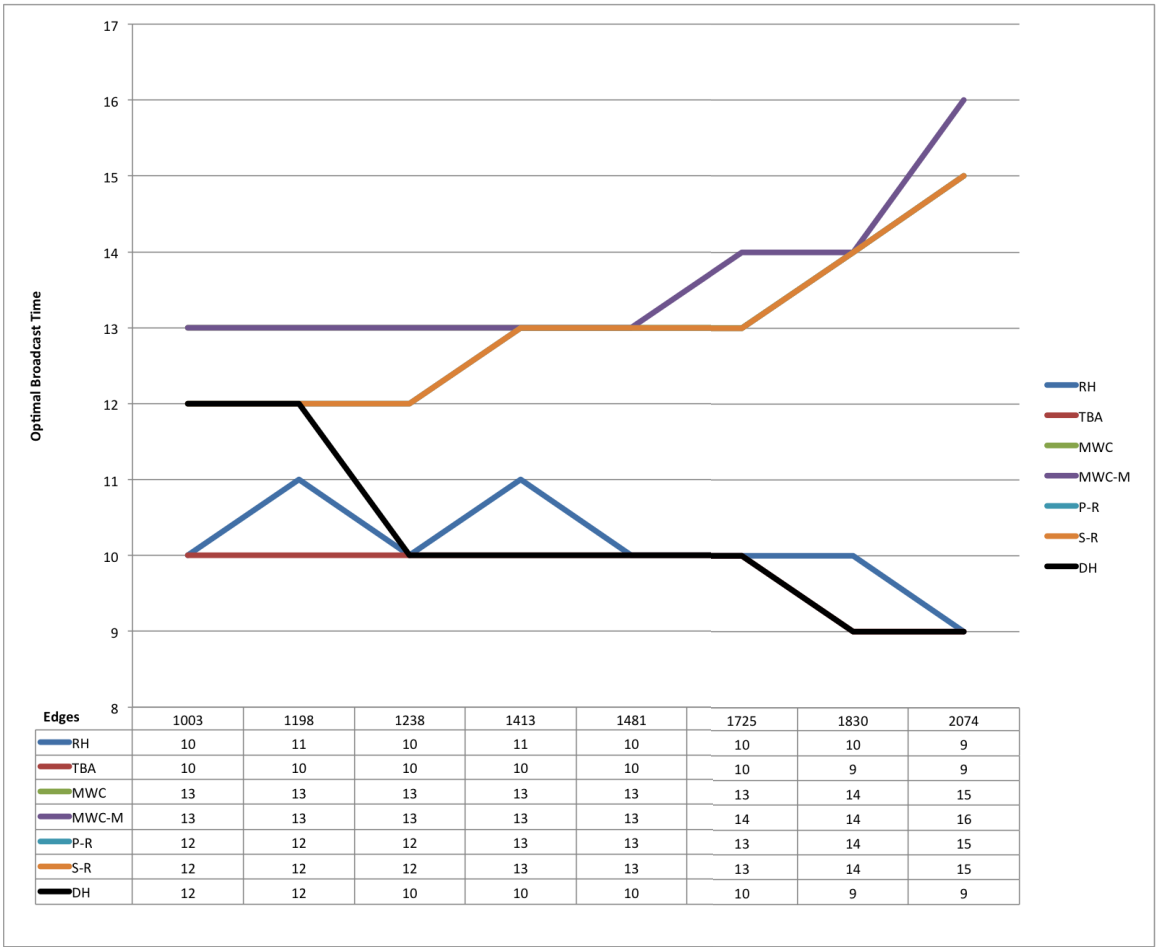


Figure 26: Chart of simulation results for GT-ITM Random model with 200 vertices

The simulation results in the GT-ITM Random model with 500 vertices are shown in Table 9 and Figure 27. In this case the Tree Based Algorithm has the best results for all range of nodes, with Deep Heuristic and the Round Heuristic following closely, whereas the results of all other previous algorithms climb up slowly as the number of edges increases. The performance of the Deep Heuristic gets better with the increase in number of edges and when the number of edges is 2074, while the Semi-Random algorithms results are almost twice those of the Round Heuristic and the Tree Based Algorithm, the performance of the Deep Heuristic gets closer to the best performing algorithm. Compared to the Round Heuristic, the Deep Heuristic has the advantage of a much lower time complexity; therefore the only previous algorithm with similar time complexity that beats the Deep heuristic for all number of edges, is TBA.

P	**Edges**	**RH**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
0.008	1003	10	10	13	13	12	12	12
0.009	1198	11	10	13	13	12	12	12
0.01	1238	10	10	13	13	12	12	10
0.011	1413	11	10	13	13	13	13	10
0.012	1481	10	10	13	13	13	13	10
0.014	1725	10	10	13	14	13	13	10
0.015	1830	10	9	14	14	14	14	9
0.016	2074	9	9	15	16	15	15	9

Table 9: Practical results for GT-ITM Random model with 500 vertices



Edges	1003	1198	1238	1413	1481	1725	1830	2074
RH	10	11	10	11	10	10	10	9
TBA	10	10	10	10	10	10	9	9
MWC	13	13	13	13	13	13	14	15
MWC-M	13	13	13	13	13	14	14	16
P-R	12	12	12	13	13	13	14	15
S-R	12	12	12	13	13	13	14	15
DH	12	12	10	10	10	10	9	9

Figure 27: Chart of simulation results for GT-ITM Random model with 500 vertices

4.2.2 GT-ITM Transit-Stub Model

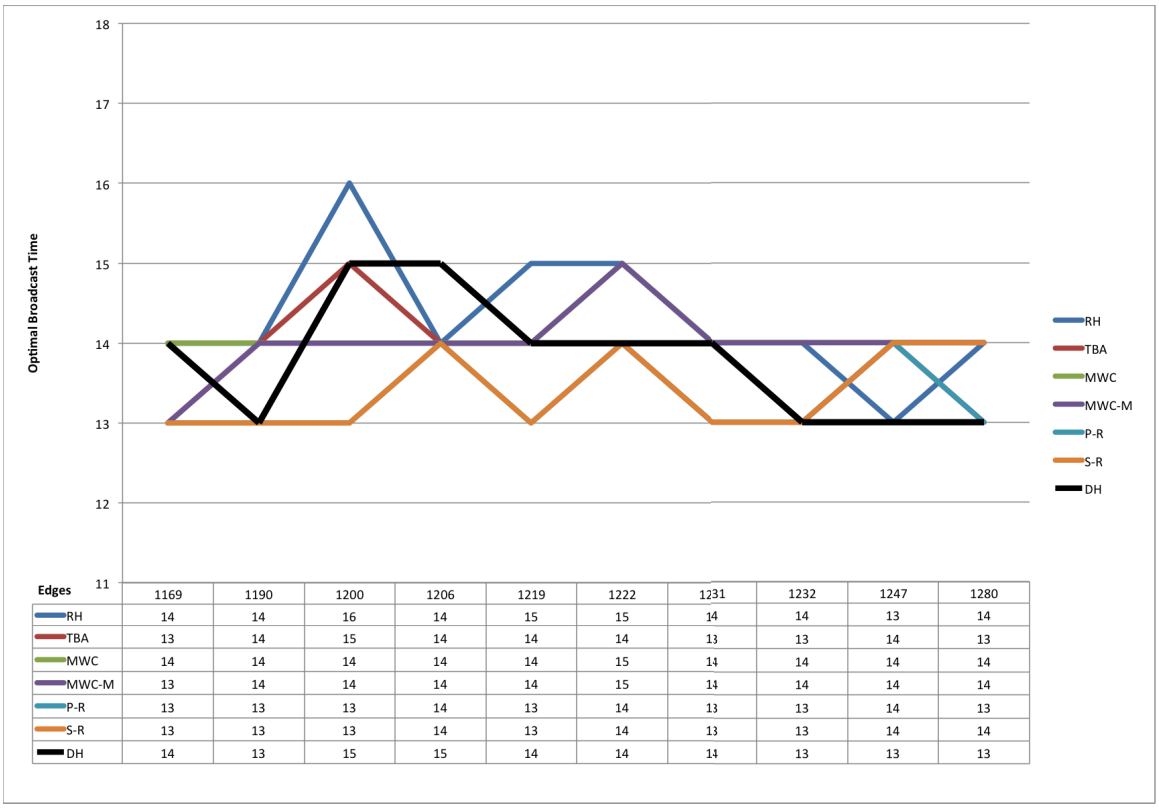
In this section, results of GT-ITM Transit-Stub Model are represented and analyzed. For the simulation, two types of GT-ITM Transit-Stub graphs are considered, one with 600 vertices and the second with 1056 vertices.

The first kind of GT-ITM Transit-Stub models that we studied are generated by the following parameters. Each graph had 3 stub domains per transit node, with no extra transit-stub or stub-stub edges. There were 3 transit domains, each of which had 8 nodes, and an edge between each pair of nodes with probability 0.5. Meanwhile, each stub domain had (on average) 8 nodes, and edge probability was also 0.5. The number of vertices is given by $3 \times 8 \times (1 + 3 \times 8) = 600$.

The simulation results in graphs with increasing number of edges are presented below in Table 10 and Figure 28. The results fluctuate a lot between all the algorithms, but they remain in a small range between 13 and 16 rounds for all number of edges. For this model, the best results are given by the Random and Semi-Random algorithms for all the cases. The Deep Heuristic also matches the best results in some cases when the number of edges is greater. In most of the other cases, the Deep Heuristic performs just one round worse than the best one. However, compared to the Semi-Random algorithm, the Deep Heuristic has the advantage that it is more reliable, producing the same results for repeated runs, whereas the results of the Semi-Random algorithm can vary between runs. Compared to the TBA algorithm, the Deep Heuristic has the advantage that approximately one half of the vertices are informed via a shortest path from the broadcast originator, while the rest of the vertices receive the message via a path at most three hops longer.

Edges	RH	TBA	MWC	MWC-M	P-R	S-R	DH
1169	14	13	14	13	13	13	14
1190	14	14	14	14	13	13	13
1200	16	15	14	14	13	13	15
1206	14	14	14	14	14	14	15
1219	15	14	14	14	13	13	14
1222	15	14	15	15	14	14	14
1231	14	13	14	14	13	13	14
1232	14	13	14	14	13	13	13
1247	13	14	14	14	14	14	13
1280	14	13	14	14	13	14	13

Table 10: Practical results for GT-ITM Transit-Stub model with 600 vertices



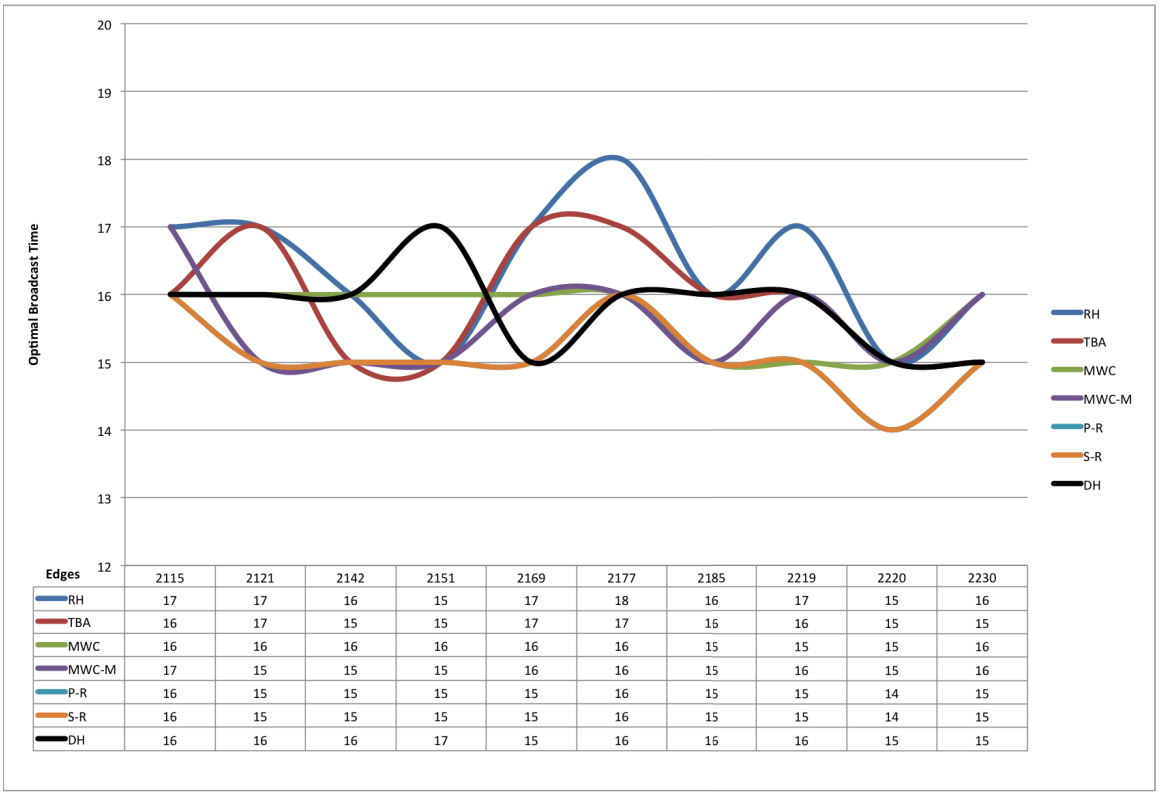
Edges	1169	1190	1200	1206	1219	1222	1231	1232	1247	1280
RH	14	14	16	14	15	15	14	14	13	14
TBA	13	14	15	14	14	14	13	13	14	13
MWC	14	14	14	14	14	15	14	14	14	14
MWC-M	13	14	14	14	14	15	14	14	14	14
P-R	13	13	13	14	13	14	13	13	14	13
S-R	13	13	13	14	13	14	13	13	14	14
DH	14	13	15	15	14	14	14	13	13	13

Figure 28: Chart of simulation results for GT-ITM Transit-Stub model with 600 vertices

Table 11 and Figure 29 show the simulation results in another kind of GT-ITM Transit-Stub model, starting with initial seed 47. Each graph has 4 stub domains per transit node, with no extra transit-stub or stub-stub edges. There are 4 transit domains, each of which has 8 nodes, and an edge between each pair of nodes with probability 0.5. Meanwhile, each stub domain has (on average) 8 nodes, and edge probability is also 0.5. Thus, the graphs have $4 \times 8 \times (1 + 4 \times 8) = 1056$ vertices.

Edges	**RH**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
2115	17	16	16	17	16	16	16
2121	17	17	16	15	15	15	16
2142	16	15	16	15	15	15	16
2151	15	15	16	15	15	15	17
2169	17	17	16	16	15	15	15
2177	18	17	16	16	16	16	16
2185	16	16	15	15	15	15	16
2219	17	16	15	16	15	15	16
2220	15	15	15	15	14	14	15
2230	16	15	16	16	15	15	15

Table 11: Practical results for GT-ITM Transit-Stub model with 1056 vertices



Edges	2115	2121	2142	2151	2169	2177	2185	2219	2220	2230
RH	17	17	16	15	17	18	16	17	15	16
TBA	16	17	15	15	17	17	16	16	15	15
MWC	16	16	16	16	16	16	15	15	15	16
MWC-M	17	15	15	15	16	16	15	16	15	16
P-R	16	15	15	15	15	16	15	15	14	15
S-R	16	15	15	15	15	16	15	15	14	15
DH	16	16	16	17	15	16	16	16	15	15

Figure 29: Chart of simulation results for GT-ITM Transit-Stub model with 1056 vertices

The results are similar to the GT-ITM Transit-Stub model with 600 vertices in the sense that the results fluctuate a lot between all the algorithms. Again, for this model, the best results are given by the Random and Semi-Random algorithms. The Deep Heuristic matches the best results in some cases and in most other cases performs just one round worse than the best one. However, compared to the Semi-Random algorithm, the Deep Heuristic has the advantage that it is more reliable, producing the same results for repeated runs, whereas the results of the Semi-Random algorithm can vary between runs. Compared to the TBA algorithm, the Deep Heuristic has the advantage that approximately one half of the vertices are informed via a shortest path from the broadcast originator, while the rest of the vertices receive the message via a path at most three hops longer.

4.2.3 Tiers Model

All the heuristics are simulated in two Tiers models, one of which has 355 vertices, while the other has 1105 vertices. The graphs of 355 vertices consist of one WAN, ten MANs and five LANs, while graphs of 1105 vertices are constituted by one WAN, ten MANs and ten LANs. All parameters used to generate these two kinds of graphs are listed in Table 12 and Table 13.

Edge	N_W	N_M	N_L	S_W	S_M	S_L	R_W	R_M	R_L	R_{MW}	R_{LM}
354	1	10	5	5	10	5	1	1	1	1	1
414	1	10	5	5	10	5	1	1	1	2	2
474	1	10	5	5	10	5	1	1	1	3	3
357	1	10	5	5	10	5	2	1	1	1	1
477	1	10	5	5	10	5	2	1	1	3	3
535	1	10	5	5	10	5	2	1	1	4	4
422	1	10	5	5	10	5	3	2	1	2	2
482	1	10	5	5	10	5	3	2	1	3	3
541	1	10	5	5	10	5	3	2	1	4	4

Table 12: Parameters for Tiers model with 355 vertices

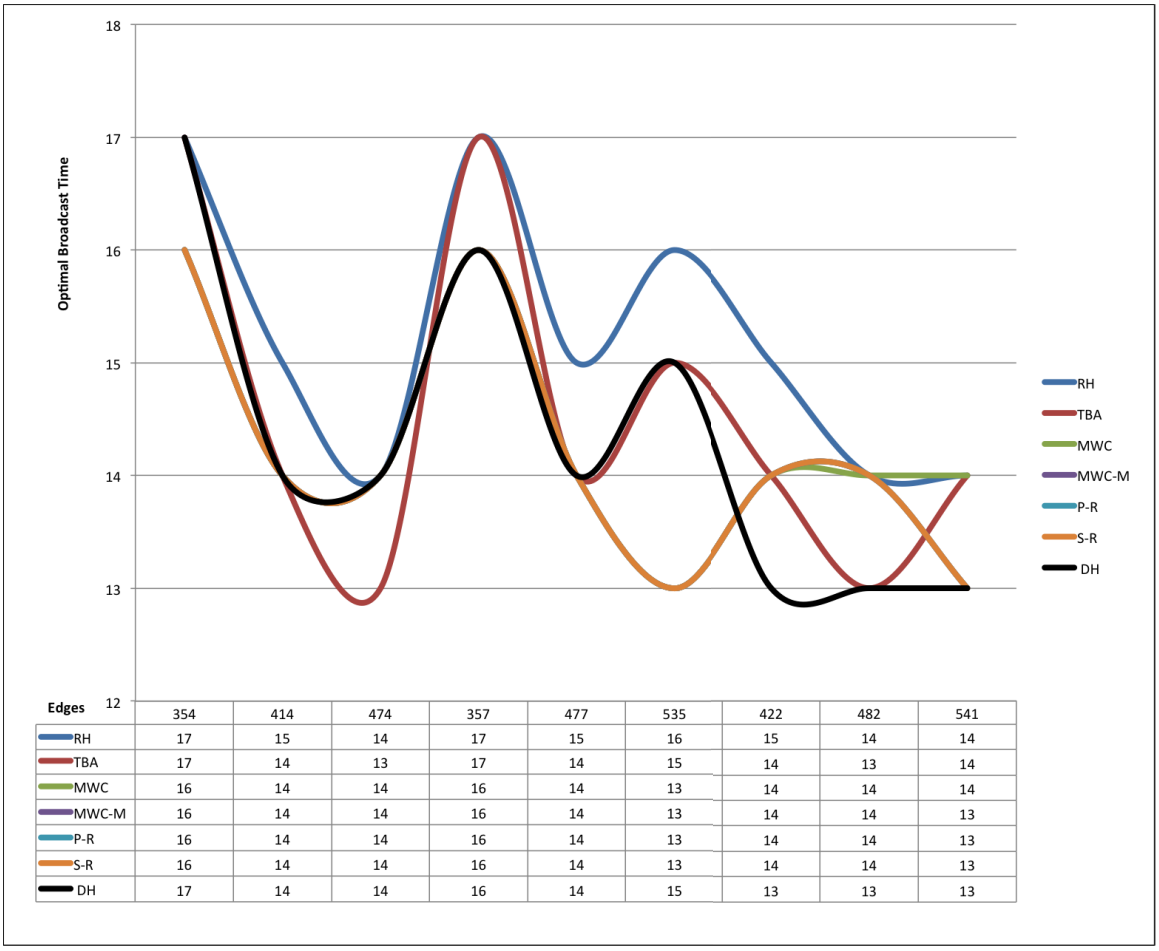
Edge	N_W	N_M	N_L	S_W	S_M	S_L	R_W	R_M	R_L	R_{MW}	R_{LM}
1214	1	10	10	5	10	10	1	1	1	2	2
1324	1	10	10	5	10	10	1	1	1	3	3
1447	1	10	10	5	10	10	1	2	1	4	4
1106	1	10	10	5	10	10	2	2	1	1	1
1216	1	10	10	5	10	10	2	2	1	2	2
1326	1	10	10	5	10	10	2	2	1	3	3
1110	1	10	10	5	10	10	3	2	1	1	1
1220	1	10	10	5	10	10	3	2	1	2	2
1331	1	10	10	5	10	10	3	2	1	3	3
1449	1	10	10	5	10	10	2	2	1	4	4

Table 13: Parameters for Tiers model with 1105 vertices

In Table 14 and Figure 30, we present the simulation results in Tiers graphs with 355 vertices and increasing number of edges from 354 to 541. We can observe that the results of all the algorithms fluctuate a lot, and it is hard to point out which one works the best. The Deep Heuristic has poor performance for low number of edges, but as the number of edges increases, its performance gets similar to the previous algorithms. Also, compared to the Semi-Random algorithm, the Deep Heuristic has the advantage that it is deterministic, producing the same results for repeated runs, whereas the results of the Semi-Random algorithm can vary between runs.

Edges	**RH**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
354	17	17	16	16	16	16	17
414	15	14	14	14	14	14	14
474	14	13	14	14	14	14	14
357	17	17	16	16	16	16	16
477	15	14	14	14	14	14	14
535	16	15	13	13	13	13	15
422	15	14	14	14	14	14	13
482	14	13	14	14	14	14	13
541	14	14	14	13	13	13	13

Table 14: Practical results for Tiers model with 355 vertices



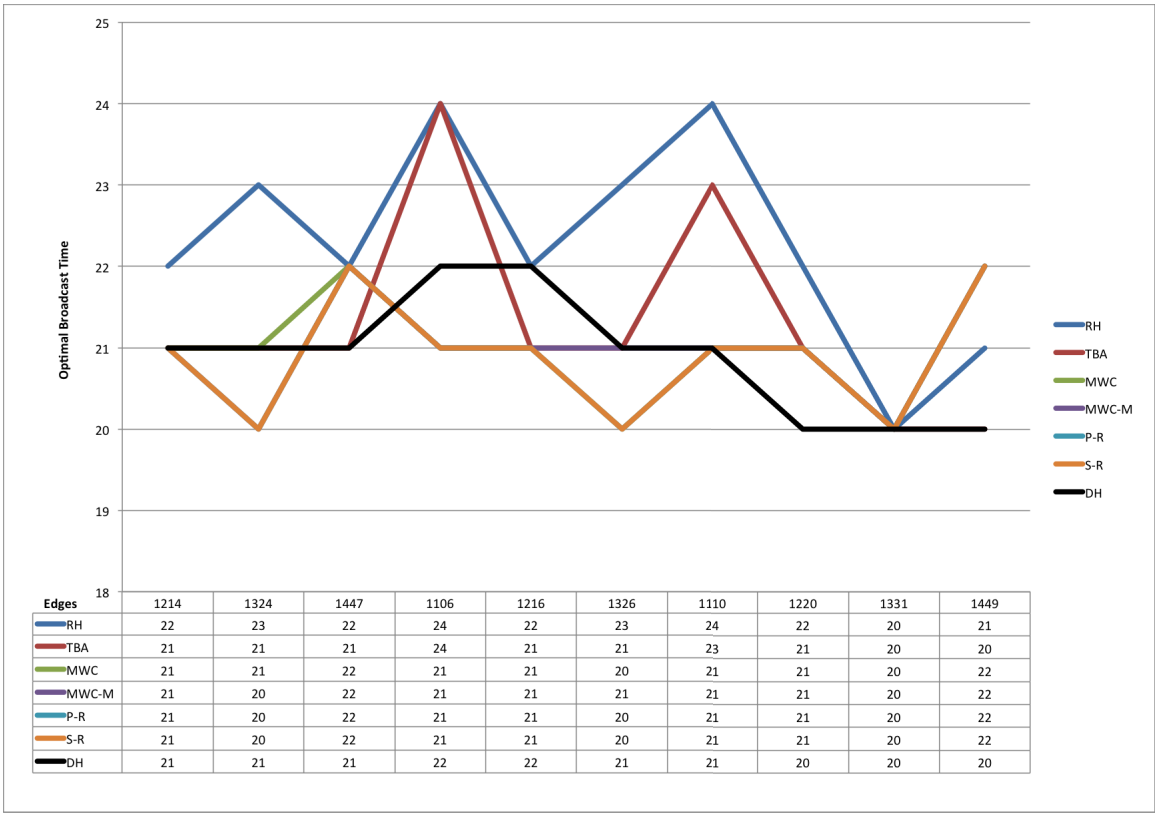
Edges	354	414	474	357	477	535	422	482	541
RH	17	15	14	17	15	16	15	14	14
TBA	17	14	13	17	14	15	14	13	14
MWC	16	14	14	16	14	13	14	14	14
MWC-M	16	14	14	16	14	13	14	14	13
P-R	16	14	14	16	14	13	14	14	13
S-R	16	14	14	16	14	13	14	14	13
DH	17	14	14	16	14	15	13	13	13

Figure 30: Chart of simulation results for Tiers model with 355 vertices

In Table 15 and Figure 31, we present the simulation results in Tiers graphs with 1105 vertices and different number of edges between 1106 and 1449. Once again, we can observe that the results of all the algorithms fluctuate a lot. The Random and Semi- Random algorithms give best results in seven of the ten graphs, and the Deep Heuristic gives the best results in three graphs, the ones with 1447, 1216, and 1220 edges. However, compared to the Semi-Random algorithm, the Deep Heuristic has the advantage that it is deterministic, producing the same results for repeated runs, whereas the results of the Semi-Random algorithm can vary between runs. Compared to the TBA algorithm, the Deep Heuristic has the advantage that approximately one half of the vertices are informed via a shortest path from the broadcast originator, while the rest of the vertices receive the message via a path at most three hops longer.

Edges	**RH**	**TBA**	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
1214	22	21	21	21	21	21	21
1324	23	21	21	20	20	20	21
1447	22	21	22	22	22	22	21
1106	24	24	21	21	21	21	22
1216	22	21	21	21	21	21	22
1326	23	21	20	21	20	20	21
1110	24	23	21	21	21	21	21
1220	22	21	21	21	21	21	20
1331	20	20	20	20	20	20	20
1449	21	20	22	22	22	22	20

Table 15: Practical results for Tiers model with 1105 vertices



Edges	1214	1324	1447	1106	1216	1326	1110	1220	1331	1449
RH	22	23	22	24	22	23	24	22	20	21
TBA	21	21	21	24	21	21	23	21	20	20
MWC	21	21	22	21	21	20	21	21	20	22
MWC-M	21	20	22	21	21	21	21	21	20	22
P-R	21	20	22	21	21	20	21	21	20	22
S-R	21	20	22	21	21	20	21	21	20	22
DH	21	21	21	22	22	21	21	20	20	20

Figure 31: Chart of simulation results for Tiers model with 1105 vertices

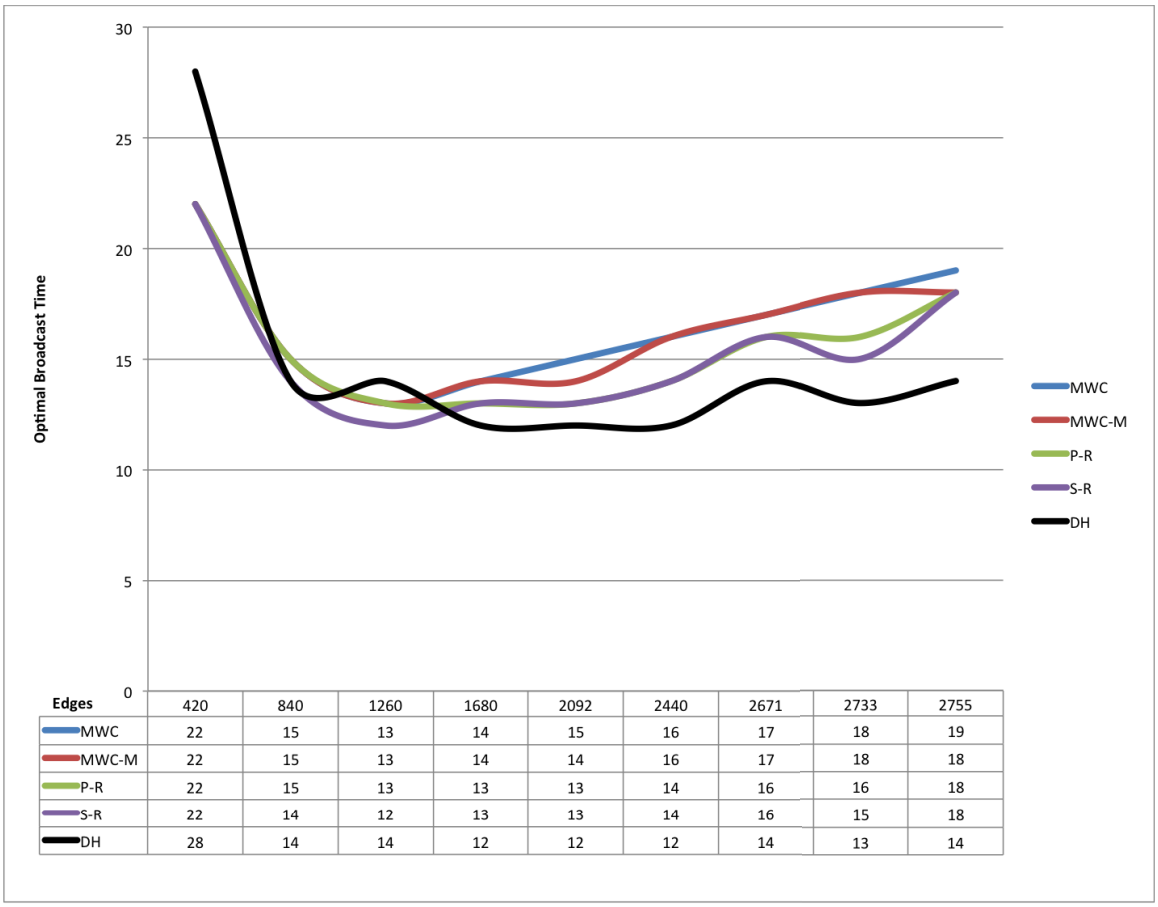
4.2.4 BRITE Top-down Hierarchical Model

In this section, simulation results of four heuristics (P-R, S-R, MWC and MWC-Modified) will be presented, since there is not any result for the other two. We simulate the heuristics in graphs with 400 and 1000 vertices, and each has Waxman and Barabasi-Albert models. The configurations of the graphs are as follows. For the graphs with 400 vertices, vertex numbers of AS-level and Route-level are both 20, the number of links added per new node ranges from 1 to 9, and the edge connection model is set to Smallest Degree. In the Waxman model, $\alpha = 0.15$, and $\beta = 0.2$. For the graphs with 1000 vertices, the vertex number of AS-level is 20, the vertex number of Route-level is 50, the number of links added per new node ranges from 1 to 9, and the edge connection model is set to Smallest Degree. In the Waxman model, $\alpha = 0.15$, and $\beta = 0.2$.

In Table 16 and Figure 32 we present the simulation results in the BRITE Top-down Waxman model with 400 vertices. With the number of edges increasing, the results of the previous four heuristics decline first, and then ascend slowly. In contrast, we can clearly observe that the Deep Heuristic not only performs better as the number of edges increases, but the difference of 6 rounds better in the graph with 2755 edges is quite significant compared to the Semi-Random algorithm for example.

Edges	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
420	22	22	22	22	28
840	15	15	15	14	14
1260	13	13	13	12	14
1680	14	14	13	13	12
2092	15	14	13	13	12
2440	16	16	14	14	12
2671	17	17	16	16	14
2733	18	18	16	15	13
2755	19	18	18	18	14

Table 16: Practical results for BRITE Top-down Waxman model with 400 vertices



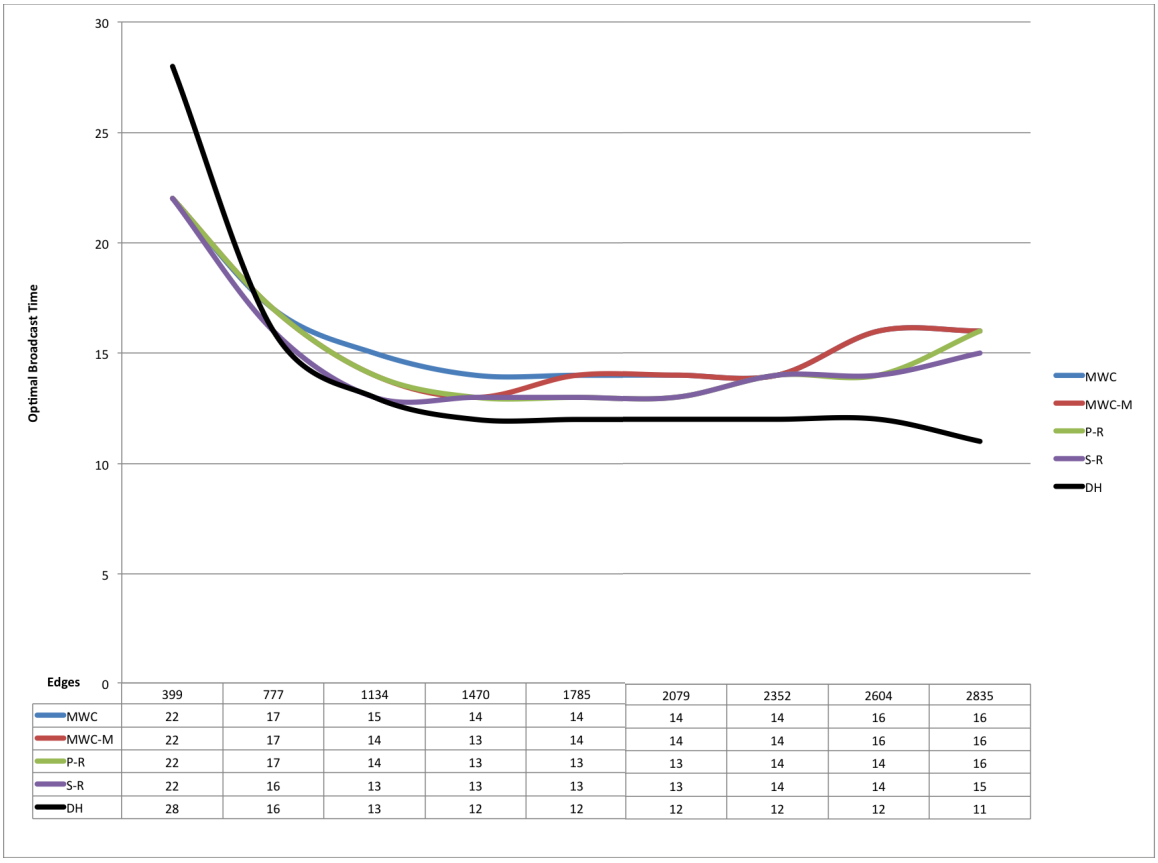
Edges	420	840	1260	1680	2092	2440	2671	2733	2755
MWC	22	15	13	14	15	16	17	18	19
MWC-M	22	15	13	14	14	16	17	18	18
P-R	22	15	13	13	13	14	16	16	18
S-R	22	14	12	13	13	14	16	15	18
DH	28	14	14	12	12	12	14	13	14

Figure 32: Chart of simulation results for BRITE Top-down Waxman model with 400 vertices

In Table 17 and Figure 33 we present the simulation results in the BRITE Top-down Barabasi-Albert model with 400 vertices. The results are similar to the previous model, the Waxman with 400 vertices. The Deep Heuristic provides the best results as the number of edges increases, whereas the performance of the previous four algorithms slowly gets worse with higher number of edges. Once again, for the graph with the most edges, 2835, the difference of 4 rounds by which the Deep Heuristic is better than the next one in performance, is quite significant.

Edges	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
399	22	22	22	22	28
777	17	17	17	16	16
1134	15	14	14	13	13
1470	14	13	13	13	12
1785	14	14	13	13	12
2079	14	14	13	13	12
2352	14	14	14	14	12
2604	16	16	14	14	12
2835	16	16	16	15	11

Table 17: Practical results for BRITE Top-down BA model with 400 vertices



Edges	399	777	1134	1470	1785	2079	2352	2604	2835
MWC	22	17	15	14	14	14	14	16	16
MWC-M	22	17	14	13	14	14	14	16	16
P-R	22	17	14	13	13	13	14	14	16
S-R	22	16	13	13	13	13	14	14	15
DH	28	16	13	12	12	12	12	12	11

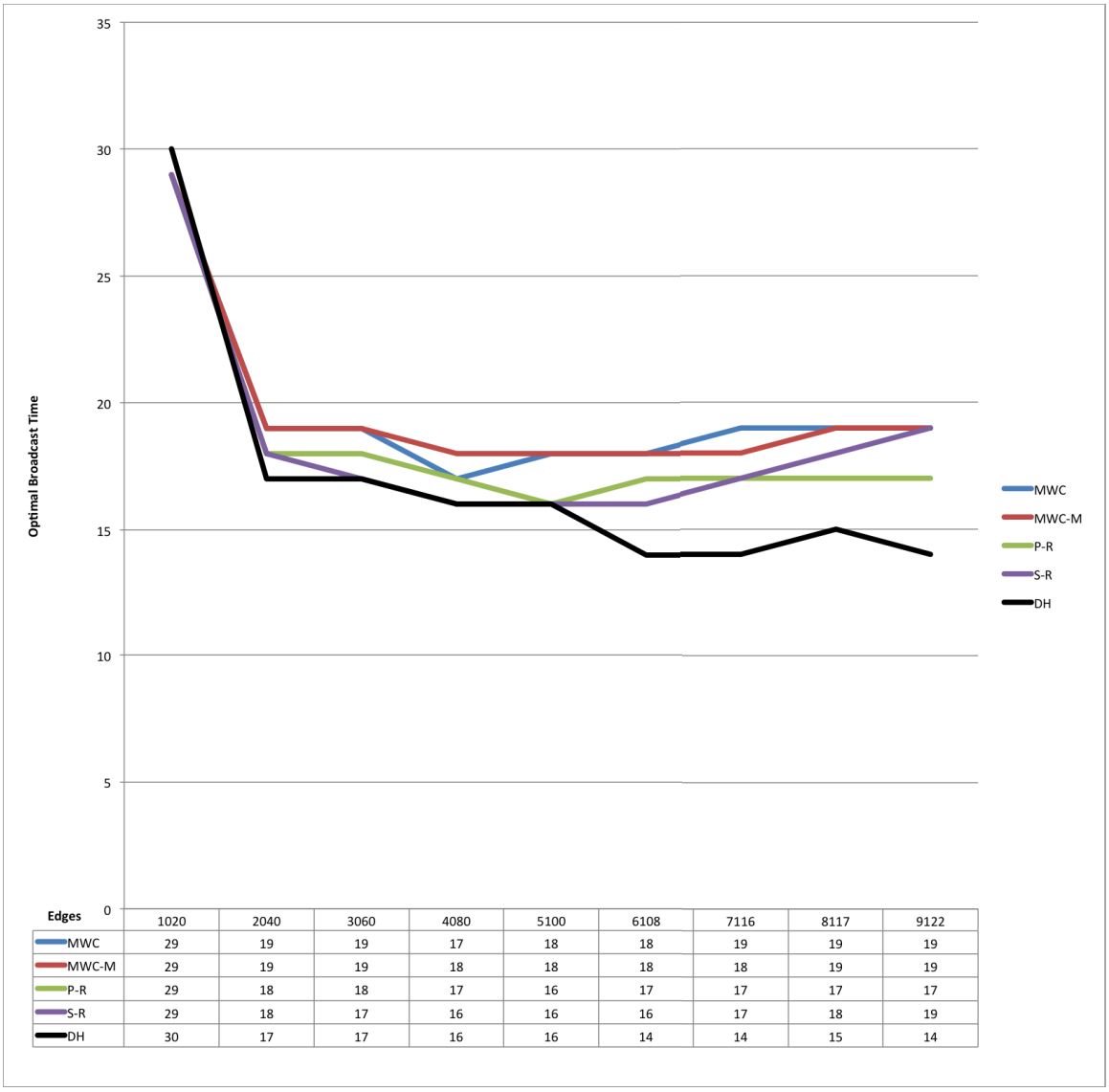
Figure 33: Chart of simulation results for BRITE Top-down BA model with 400 vertices

In the BRITE Top-down models with 1000 vertices the number of vertices at AS-level is 20, and at Route-level is 50, the number of links added per new node ranges from 1 to 9, and the edge connection model is set to Smallest Degree. The parameters of the Waxman model are $\alpha = 0.15$, and $\beta = 0.2$.

In Table 18 and Figure 34 we present the simulation results in the BRITE Top-down Waxman model with 1000 vertices. The behavior of the algorithms studied is similar to the behavior in the models with 400 vertices. The previous algorithms exhibit decreasing performance with the increase in number of edges. In contrast the performance of the Deep Heuristic improves and for the graphs with high number of edges it gets 2 or 3 rounds better than the next best results.

Edges	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
1020	29	29	29	29	30
2040	19	19	18	18	17
3060	19	19	18	17	17
4080	17	18	17	16	16
5100	18	18	16	16	16
6108	18	18	17	16	14
7116	19	18	17	17	14
8117	19	19	17	18	15
9122	19	19	17	19	14

Table 18: Practical results for BRITE Top-down Waxman model with 1000 vertices



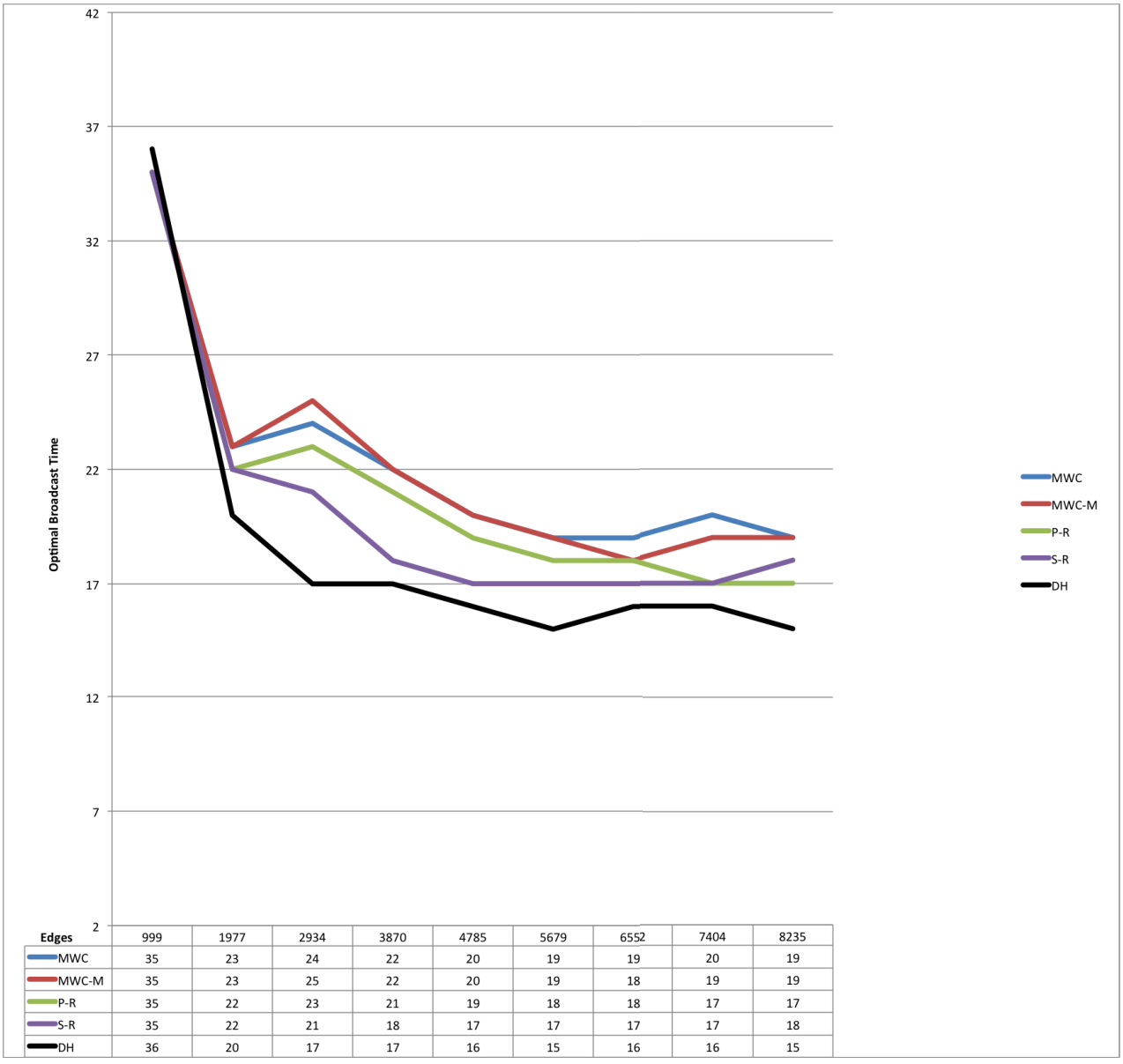
Edges	1020	2040	3060	4080	5100	6108	7116	8117	9122
MWC	29	19	19	17	18	18	19	19	19
MWC-M	29	19	19	18	18	18	18	19	19
P-R	29	18	18	17	16	17	17	17	17
S-R	29	18	17	16	16	16	17	18	19
DH	30	17	17	16	16	14	14	15	14

Figure 34: Chart of simulation results for BRITE Top-down Waxman model with 1000 vertices

In Table 19 and Figure 35 we present the simulation results in the BRITE Top-down Barabasi-Albert model with 1000 vertices. The trend we observed in the previous BRITE models appears also in this model. With the exception of the graph with 999 edges, the Deep Heuristic beats the next best results of the previous algorithms by at least 2 rounds, and in a couple of graphs by 4 rounds.

Edges	**MWC**	**MWC-M**	**P-R**	**S-R**	**DH**
999	35	35	35	35	36
1977	23	23	22	22	20
2934	24	25	23	21	17
3870	22	22	21	18	17
4785	20	20	19	17	16
5679	19	19	18	17	15
6552	19	18	18	17	16
7404	20	19	17	17	16
8235	19	19	17	18	15

Table 19: Practical results for BRITE Top-down BA model with 1000 vertices



Edges	999	1977	2934	3870	4785	5679	6552	7404	8235
MWC	35	23	24	22	20	19	19	20	19
MWC-M	35	23	25	22	20	19	18	19	19
P-R	35	22	23	21	19	18	18	17	17
S-R	35	22	21	18	17	17	17	17	18
DH	36	20	17	17	16	15	16	16	15

Figure 35: Chart of simulation results for BRITE Top-down BA model with 1000 vertices

4.3 Conclusion based on practical simulation

Based on the practical simulation of the Deep Heuristic in various network models as well as network topologies, we can understand where the use Deep Heuristic would be most appropriate and where it lags behind other known algorithms.

Commonly Used Topologies

We first examined the commonly used topologies like Hypercube (H_d), Cube Connected Cycles (CCC_d), Shuffle-Exchange (SE_d), deBruijn (DB_d) and Butterfly (BF_d). The results of Deep Heuristics were compared with known results of other algorithms.

- In Hypercube (H_d), we saw that the Deep Heuristic performed exactly same as TBA, resulting it to be the best performing algorithm in all dimensions.
- In Cube Connected Cycles (CCC_d), we saw that on average, the Deep Heuristic took 1 time unit less than other algorithm except TBA, which was performing same as Deep Heuristic. Although, for dimension 20, Deep Heuristic found to be taking even one less time unit than TBA.
- In Shuffle-Exchange (SE_d), Deep Heuristic performed same as TBA up to 8 dimensions, it performed same as P-R up to dimension 15, where it started to be performing same as TBA, making them best performing algorithms.
- In deBruijn (DB_d), the Deep Heuristic found to be performing same as TBA up to dimension 13, and after 13 dimensions, DH found to be taking 1 round less than TBA because it was choosing an optimal sub-tree while performing the broadcast operation.
- In Butterfly (BF_d), we saw that the Deep Heuristic was performing same as TBA up to dimension 11 and after that, it found to be performing taking one round less than TBA, making it best performing algorithm amongst all.

So, in the Commonly Used Topologies, DH found to be performing best in most of the cases as the dimension of the graph was increasing.

NS-2 Models

We examined other network models like GT-ITM Random model, GT-ITM Transit-Stub Model, Tiers Model and BRITE Top-down Hierarchical Model.

- In GT-ITM Random model, with small number of edges (small P), the Deep heuristic performs slightly worse than previous heuristics because of some precalculations in the earlier stages, but as the number of edges increases, the Deep Heuristic tends to perform better than all other heuristics.
- In GT-ITM Transit-Stub Model, when the number of edges are around 1247 to 1280 with 600 vertices, DH seemed to be performing best, but as the number of vertices increased to 1056, it was no longer performing best in all the cases.
- In Tiers model, DH found to be performing not so good for less number of edges, but as the number of edges grew, the number of rounds started to decrease.
- In all variants of BRITE Top-down Hierarchical model, DH found to be performing best amongst all other heuristics.

Chapter 5

Conclusion and Future Work

In the modern world where every industry sector depends on connectivity with the rest of the world, it is very important to have faster ways to transmit information with minimum latency possible. Broadcasting in graphs is one of the elementary communication primitives, which means that having a sophisticated and efficient broadcast scheme is very helpful to achieve the goal of higher performance in parallel systems and as a result, in network connectivity.

The primary focus of this thesis has been to design and optimize algorithms that can generate efficient broadcast time for any given arbitrary graph. However, it is a proven fact the determination of broadcast time for an arbitrary vertex u in an arbitrary graph G is NP-complete. Hence, we surveyed existing approximation and heuristic algorithms and analyzed their behavior in commonly used topologies and other topologies used to study networking algorithms.

We designed an efficient heuristic, which improves behavior of some existing heuristics in certain key situations. As a result, for some cases, our new heuristic makes the calculation even faster than best-known heuristics like TBA, Round Heuristic, etc. The new heuristic analyzes the graph on each level before deciding where to send the information and that leads us towards improvement in broadcast time. It was found that one of the important properties, which determines strength of a subgraph to participate in broadcasting was, the density of the subgraph. The new heuristic

focuses on that property and generates the broadcast tree for the subgraphs and makes the selection of subgraph, that would result in a better broadcast scheme. To do so, the new heuristic examines the sub-graphs deeply to eliminate the edges that are not going to be helpful for broadcasting, which would result into a lower estimated broadcast time and hence, which would result in a more efficient way of generating the broadcasting scheme. That is why the new heuristic is named Deep Heuristic.

By observing their performance in the commonly used topologies and network models, we see that the new heuristic is very well suitable for graphs where most of the vertices have high degree and higher density. Based on our extensive simulations, we conclude that the Deep Heuristics perform exceptionally well in some of the models representing real networks. In BRITE Top-down hierarchical model topologies the results are much better than previous heuristics. The Deep Heuristic not only gives the best results, but it consistently beats the best previous heuristics by two or more rounds. In GT-ITM models it is similar to previous heuristics.

Time complexity is another essential benchmark to evaluate the performance of an algorithm. The Deep heuristic has a time complexity of $O(|E| + |V| \log |V|)$, which is lower than that of many other heuristics mentioned in this thesis. The low time complexity helps to generate broadcast schemes for large graphs, and to obtain new upper bounds on the broadcast time.

The research in this thesis can be continued in several directions. For the heuristics, the matching strategy could be improved so that better results might be achieved. Furthermore, the implementation of the heuristics could be optimized, and ultimately construct a Breadth First Search tree that generates the minimum broadcast time. On the other hand, another direction is mainly based on the layer graph. First, instead of a heuristic, an approximation algorithm could be designed for the broadcast time problem with constant number of layers, or even in the graph of variable layers. Second, the lower and upper bounds on the broadcast time obtained from the layer graph could be another interesting research direction.

Bibliography

[1] A. Bar-Noy, J. Bruck and C. T. Ho and S. Kipnis and B. Schieber. Computing global combine operations in the multi-port postal model. *IEEE Trans. Par. Distr. Syst.*, 6:896–900, 1995.

[2] A.M. Farley, S. Hedetniemi, S. Mitchell, and A. Proskurowski. Minimum broadcast graphs. *Discrete Mathematics*, 25:189–193, 1979.

[3] R. Beier and J F. Sibeyn. A powerful heuristic for telephone gossiping. In *Proceedings of the 7th International Colloquium on Structural Information and Communication Complexity (SIROCCO-00)*, pages 17–36, L'Aquila, Italy, 2000.

[4] Jean-Claude Bermond and C Peyrat. Broadcasting in de Bruijn networks. In *Proc. 19th SE Conference on Combinatorics, Congressus Numerantium*, volume 66, pages 283–292, 1988.

[5] Calvert, Kenneth L and Doar, Matthew B and Zegura, Ellen W. Modeling internet topology. *Communications Magazine, IEEE*, 35(6):160–163, 1997.

[6] L. Changhong and Z. Kemin. The broadcast function value b(23) is 33 or 34. *Acta Mathematicae Applicatae Sinica (English Series)*, 16(3):329–331, 2000.

[7] D. W. Krumme, George Cybenko, K. N. Venkataraman. Gossiping in minimal time. *SIAM J. Computing*, 21(1):111–139, 1992.

[8] M.B. Doar. A better model for generating test networks. *Global Telecommunications Conference*, IEEE:86–93, 1996.

[9] E.W. Zegura, K.L. Calvert and S. Bhattacharjee. How to model an internetwork, in INFOCOM'96. *Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, pages 594–602, 1996.

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

[11] A.M. Farley and S.T. Hedetniemi. Broadcasting in grid graphs. In *In Proceedings of the 9th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 275–288, 1978.

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

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

[14] G. Barsky, H. Grigoryan, and H.A. Harutyunyan. Lower bounds on broadcast function for n = 24 and 25. *Discrete Applied Mathematics*, 175:109–114, 2014.

[15] M.R. Gary and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman and Company, New York, 1979.

[16] M. Grigni and D. Peleg. Tight bounds on minimum broadcast networks. *SIAM j. Discr. Math.*, 4:207–222, 1991.

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

[18] H. A. Harutyunyan and B. Shao. A heuristic for k-broadcasting in arbitrary networks. In *IEEE Conference on Applications of Graph Theory IV03*, pages 287–292. London, England, 2003.

[19] H. A. Harutyunyan and B. Shao. Optimal k-broadcast in trees. *Congressus Numerantium*, 64, 2003.

[20] H.A. Harutyunyan. An efficient vertex addition method for broadcast networks. *Internet Mathematics*, 5(3):211–225, 2008.

[21] H.A. Harutyunyan and B. Shao. An efficient heuristic for broadcasting in networks. *Parallel and Distributed Computing*, 66(1):68–76, 1981.

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

[23] S.M Hedetniemi, S.T Hedetniemi, and A.L Liestman. A surway of gossiping and broadcasting in comminication networks. *Networks*, 18:319–349, 1996.

[24] Hromkovič, Juraj and Klasing, Ralf and Monien, Burkhard and Peine, Regine. Dissemination of information in interconnection networks (broadcasting & gossiping). In *Combinatorial network theory*, pages 125–212. Springer, 1996.

[25] J. Hromkovic, R. Klasing and B. Monien and R. Peine. *Combinatorial Network Theory*. Kluwer Academic Publishers, 1996.

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

[27] L.H. Khachatrian and O.S. Harutounian. Construction of new classes of minimal broadcast networks. *Conference on Coding Theory*, pages 69–77, 1990.

[28] W. Knödel. New gossips and telephones. *Discrete Mathematics*, 13(1):95, 1975.

[29] J.-C. König and E.Lazard. Minimum k-broadcast graphs. *Discr. Appl. Math.*, 53:199–209, 1994.

[30] R. Labahn. A minimum broadcast graph on 63 vertices. *Discrete Applied Mathematics*, 53(1-3):247–250, 1994.

[31] E. Lazard. Broadcasting in dma-bound bounded degree graphs. *Discr. Appl. Math.*, 37/38, 1992.

[32] S. Lee. *Information dissemination theory in communication networks: Design of c Broadcast Networks*. PhD thesis, Pennsylvania State University, 1999.

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

[34] A.L. Liestman and J.G. Peters. Broadcast networks of bounded degree. *SIAM Journal on Discrete Mathematics*, 1(4):531–540, 1988.

[35] M. Dinneen, M. Fellows and V. Faber. Algebraic constructions of efficient broadcast networks. *Applied Algebra, Algebraic Algorithms and Error- Correcting Codes*, 539:152–158, 1991.

[36] M. Maheo and J.F. Sacle. Some minimum broadcast graphs. *Discrete Applied Mathematics*, 53(1-3):285, 1994.

[37] Medina Alberto, Lakhina Anukool, Matta Ibrahim, Byers John. Brite: An approach to universal topology generation. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, pages 346–353. IEEE, 2001.

[38] V.E. Mendia and D. Sarakar. optima broadcasting on the star graph. *IEEE Trans. Parallel Distrib. System*, 3:389–396, 1992.

[39] S. Mitchell and S. Hedetniemi. A census of minimum broadcast graphs. *Journal of Combinatorics, Information and System Sciences*, 5:141–151, 1980.

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

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

[42] Diestel. R. *Graph Theory*. Springer-Verlag, 3^{rd} edition, 2005.

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

[44] R. Ravi. Rapid rumor ramification: approximating the minimum broadcast time. *Pro- ceedings of 35th Annual Symposium on Foundation of Computer Sciencet*, pages 202–213, 1994.

[45] J.F. Sacle. Lower bounds for the size in four families of minimum broadcast graphs. *Discrete Mathematics*, 150(1):359–369, 1996.

[46] P. Scheuermann and M. Edelberg. Optimal broadcasting in point-to-point computer networks. *Technical Report*, 1981.

[47] P. Scheuermann and G. Wu. Heuristic algorithms for broadcasting in point-to-point computer network. *IEEE Transactions on Computers*, C33(9), 1984.

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

[49] A. Shastri and S. Gaur. Multi-broadcasting in communication networks. In *Proc. Int. Symp. on Communications (ISCOM97)*, pages 167–170. 1997.

[50] F. Chung W. Aiello and L. Lu. Random evolution in massive graphs. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 510–519, 2001.

[51] Wang, Wei. *Heuristics for Message Broadcasting in Arbitrary Networks.* PhD thesis, Concordia University, 2010.

[52] Waxman, Bernard M. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622, 1988.

[53] J. Xiao and X. Wang. A research on minimum broadcast graphs. *Chinese Journal of Computers*, 11:99–105, 1988.

[54] Zegura, Ellen W and Calvert, Kenneth L and Donahoo, Michael J. A quantitative comparison of graph-based models for internet topology. *IEEE/ACM Transactions on Networking (TON)*, 5(6):770–783, 1997.

[55] J. Zhou and K. Zhang. A minimum broadcast graph on 26 vertices. *Applied Mathematics Letters*, 14(8):1023–1026, 2001.