New Heuristic for Message Broadcasting in Arbitrary Networks

Cosmin Jimborean

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 at

Concordia University

Montreal, Quebec, Canada

September 2013

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Cosmin Jimborean

Entitled: New 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 regulation of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

____ Chair

Dr. N. Shiri

____ Examiner

Dr. J. W. Atwood

____ Examiner

Dr. D. Goswami

____ Supervisor

Dr. H. Harutyunyan

Approved by ____

Chair of Department or Graduate Program Director

________2013 ____

Dr. Christopher W. Trueman, Interim Dean

Faculty of Engineering and Computer Science

Abstract

New Heuristic for Message Broadcasting in Arbitrary Networks

Cosmin Jimborean

Efficient information dissemination in interconnection networks is a key research area because of the major role it plays in the modern interconnected world. A vast number of topics ranging from distributed computing to Internet communication rely on efficient information dissemination. Broadcasting is one of the information dissemination primitives. The minimum broadcast time problem in arbitrary networks has been examined since the 1970s. Finding an optimal broadcasting scheme for any originator in an arbitrary network has been proved to be an NP-Hard problem. In the current thesis, a new heuristic that generates broadcast schemes in arbitrary networks is presented. The heuristic has $O(|E|\, log|V|)$ time complexity, where V is the set of nodes and E is the set of the links of the network. Computer simulations in some commonly used topologies and network models show that compared to the existing heuristics the new heuristic shows better performance in some network models, and comparable performance in other network models, while having a low complexity similar to the best existing heuristics. Another advantage of the new heuristic is that approximately one half of the vertices receive the message 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.

Acknowledgements

I would like to express my sincere thank you to my supervisor Dr. H. A. Harutyunyan, for his assistance and constructive criticism in the preparation of this thesis.

I would also like to thank my parents for their unconditional support not only throughout this degree, but also throughout all my life.

Last but not the least, I would like to thank my wife Ana who has always supported and encouraged me with unending love and respect.

Contents

List of Figures

List of Tables

1 Introduction

In computer science, information dissemination encapsulates a set of problems related to the distribution of information within an interconnection network. One of the most researched computer science topics in the last years, efficient information dissemination is important for an increasing number of topics, such as parallel and distributed computing, internet networks, virtual social networks and telecommunication networks.

In the past, computer processing power was consistently increased by boosting processor clock speed from kilohertz to gigahertz. Recently, the increases of clock speed seemed to have reached their limit and with the cost of hardware decreasing dramatically, the trend to increase processing power is to build massive multi-processor systems. The size of multi-processor systems has exploded in the last few years and hundred-thousand processor systems have already been built. Under these conditions, the problems of information dissemination are especially important when designing such massive interconnection networks for parallel and distributed computing. In parallel and distributed computing, the ability of the processors in the interconnection network to communicate efficiently is crucial. To study these problems, an interconnection network is modeled as a connected undirected graph where processors are represented by the nodes of the graph and the communication links are represented by the edges of the graph. [24]

Information dissemination includes problems related to broadcasting, accumulation and gossiping. The distribution of information from one to all is known as broadcasting. Gathering the information from all to one is known as accumulation. Gathering the information from all and distributing it to all is known as gossiping.

In this thesis we will focus on broadcasting.

1.1 Problem statement

The distribution of information can be classified in four main classes:

- Routing, distribute information from one to one;
- Broadcasting, distribute information from one to all;
- Multicasting, distribute information from one to multiple, but not all;
- Gossiping, distribute information from all to all.

In this thesis, we will focus on broadcasting, in which messages are distributed from one node to the rest of the nodes in the network.

As mentioned above, an interconnection network is modeled as a connected undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges in the graph G. Using this model, Hromkovic et al. [22] gives the following abstract definition of broadcasting

"Let $G = (V, E)$ be a graph and let $v \in V$ be a node of G. Let v know a piece of information $I(v)$ which is unknown to all nodes in $V \setminus \{v\}$. The problem is to find a communication strategy (algorithm) such that all nodes in G learn this piece of information $I(v)$."[22]

A communication strategy is a sequence of steps called rounds. A round is equivalent to one discrete time unit. The broadcasting communication strategy always starts with one given informed vertex, also called the originator, and all other vertices being uninformed. During each round, each informed vertex sends the piece of information to exactly one of its uninformed neighbor vertices. The process repeats until all vertices are informed.

It is obvious that for a given graph and a given originator, multiple broadcasting communication strategies exist. The efficiency of a communication strategy is measured by the number of communication rounds needed to distribute the information from the source vertex to all vertices.

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 [37], Slater et al. present the proof 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 [17]. The 3DM problem is reduced to the broadcast problem in polynomial time. Below we present the proof given in [37].

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 $V_0, E_1, V_1, E_2, V_2, \dots, E_k, V_k$ where $V_i \subseteq V$, $E_i \subseteq E (1 \leq i \leq k)$, $E_i = \big\{\{u, v\},\ u \in V_{i-1},\ v \notin V_{i-1}\big\}$, $V_i = V_{i-1} \cup \{v\}$, 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 $|V_0| = 1$, 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 $X = \{x_1, x_2, \dots, x_m\}$, $Y = \{y_1, y_2, \dots, y_m\}$, $Z = \{z_1, z_2, \dots, z_m\}$ and $M \subseteq X \times Y \times Z$, does there exist a subset $N \subseteq M$ and $|N| = m$, such that every two elements in N disagree in all three coordinates?[17]

Starting from the sets X, Y, Z and M in the 3DM problem, a graph G is constructed in polynomial time m as shown in Figure 1, adapted from [37]. First, each vertex $(x_i, y_j, z_k) \in M$ is connected to vertices x_i of X, y_j 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 sets 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.

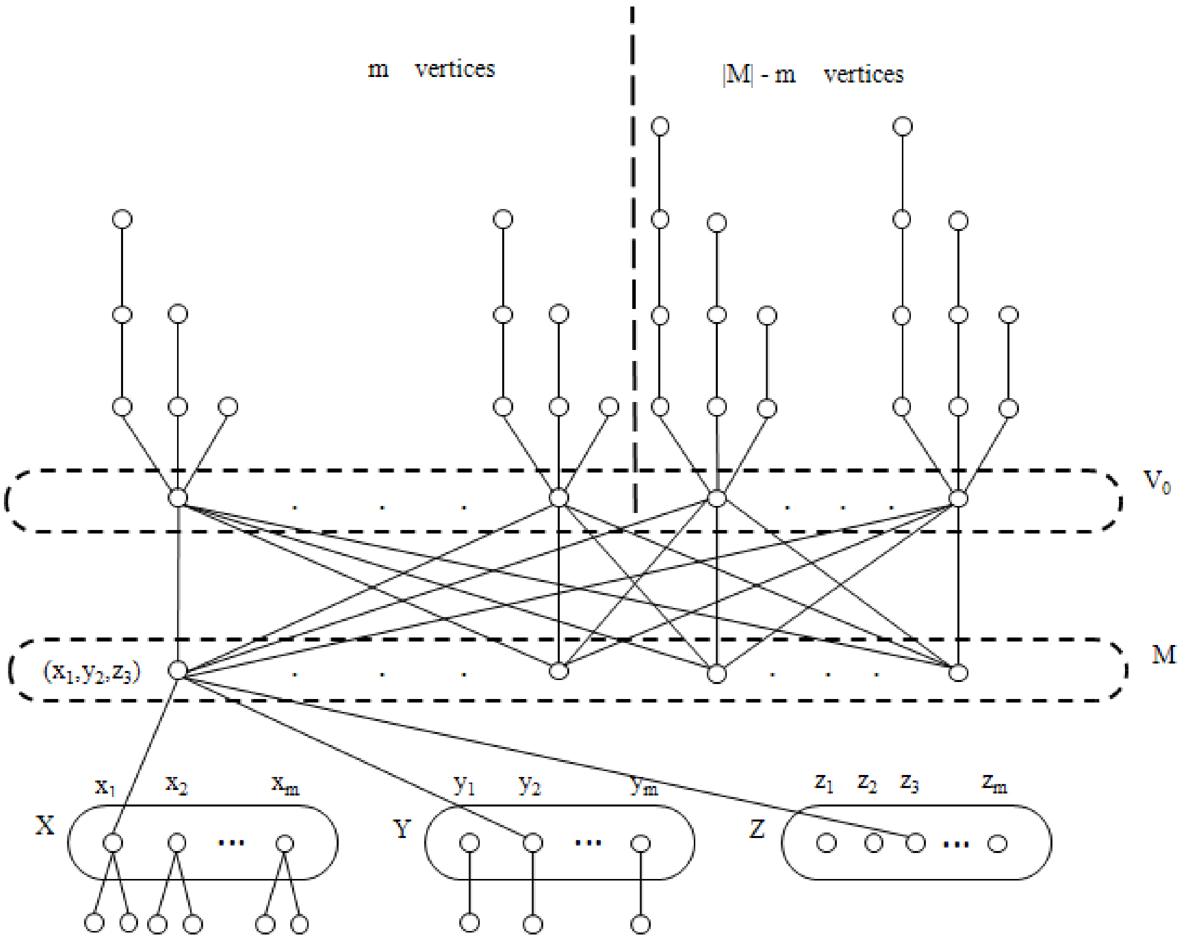


Figure 1 The graph G

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 S of M at round 1 and the vertices in S 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 S 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, adapted from [37], as follows. Starting from graph G, add a vertex u, an independent vertex set $U = \{u_1, u_2, \dots, u_m\}$, and the edges $\{(u, u_i), 1 \leq i \leq m = |V_0|\}$. Every vertex u_i joins $m - i$ paths of lengths $6, 7, \dots, m + 5 - i$. Finally, create a matching between U and V_0 by adding m edges.

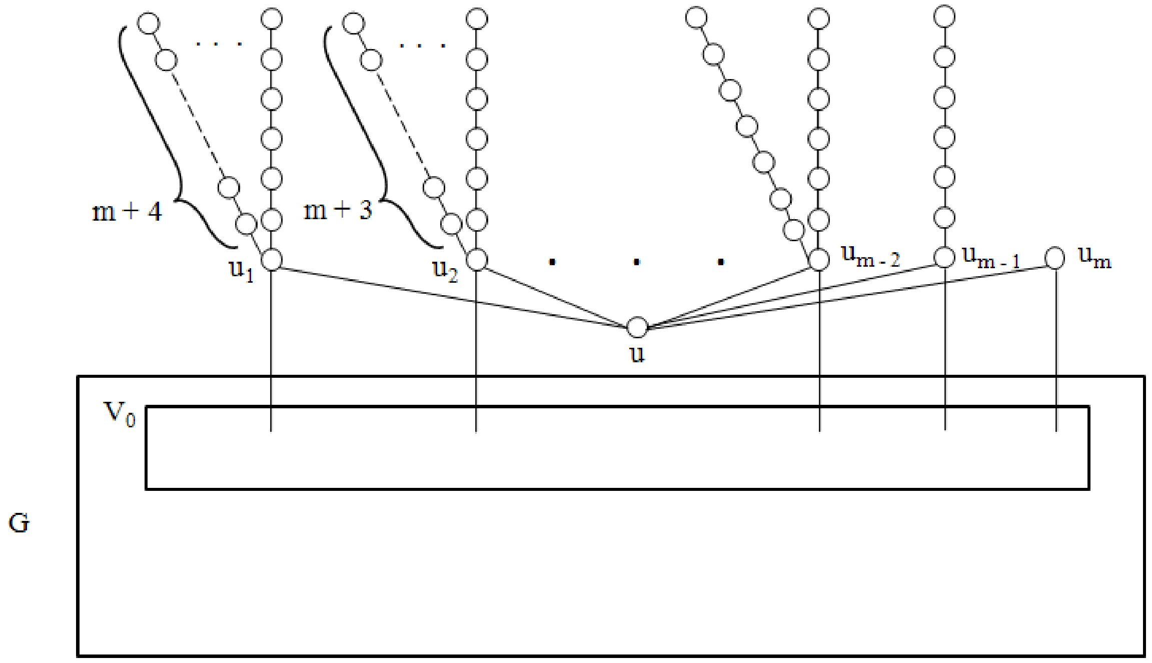


Figure 2 The Graph H

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 [25, 26], and bounded degree graphs [7, 9, 32]. In addition, researchers usually also approach the problem with approximation algorithms. Schindelhauer et al. [36] provide 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.

1.3 Thesis Outline

Chapter 1 introduces the information dissemination topic, narrowing on the broadcasting problem and the NP-completeness of this problem. Chapter 2 goes deeper into the broadcasting problem in computer networks, presenting commonly used network topologies and some of the existing broadcast heuristics. A new heuristic is presented in Chapter 3 and in Chapter 4 the simulation results of this new heuristic are compared to the simulation results of some existing heuristics.

Finally, conclusions and future work are discussed in Chapter 5.

2 Background

An interconnection topology or network topology describes the structure of a network and of the elements that compose the network, such as nodes and edges [40]. Many topologies of interconnection networks exist; in the first part of this chapter we will present a set of commonly used topologies and their properties.

2.1 Commonly Used Topologies

Commonly used topologies are topologies of interconnection networks with specific properties. They are very popular in the research community and their properties and broadcasting behavior have been studied extensively [15, 24, 29] and [20].

The Path P_n is a very simple graph composed of a sequence of n vertices where each vertex is connected to the next vertex in the sequence. In a path P_n with n vertices, the start and end vertices have degree 1, whereas all other vertices have degree 2. The diameter of path P_n is equal to $n - 1$. The broadcast time of path P_n is also $n - 1$ and is given by the broadcast time of the start and end vertices. Figure 3 shows a path with 7 vertices.

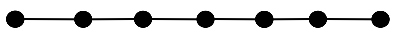


Figure 3 Path P_7

The Cycle C_n is a path P_n with the start and end vertices connected. In a cycle C_n, all vertices have degree 2, the diameter is $\left\lfloor \frac{n}{2} \right\rfloor$, and the broadcast time of the cycle is $\left\lceil \frac{n}{2} \right\rceil$. Figure 4 shows a cycle with 7 vertices.

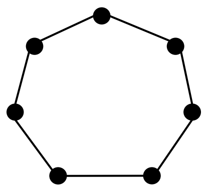


Figure 4 Cycle C_7

The Complete Graph K_n is a graph with n vertices where every two distinct vertices are connected. Figure 5 shows a complete graph with 6 vertices.

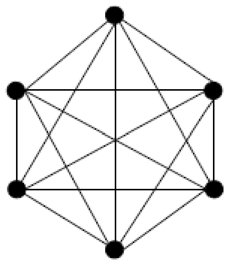


Figure 5 Complete Graph K_6

In a complete graph K_n, all vertices have degree $n-1$, the diameter is 1, and the broadcast time is $\lceil \log \mathrm{n} \rceil$. The broadcast time is obtained by noticing that at each round except the last round the number of informed vertices will double because every informed vertex will inform an uninformed neighbor.

The Hypercube H_m is a graph with 2^m vertices, where each vertex represents a binary string of length m and each vertex is connected to those vertices whose binary string representation differs in exactly one bit. An $(m + 1)$-dimensional hypercube can be constructed by connecting each pair of corresponding vertices of two m-dimensional hypercubes. Figure 6 shows two hypercubes of dimension 3 and 4.

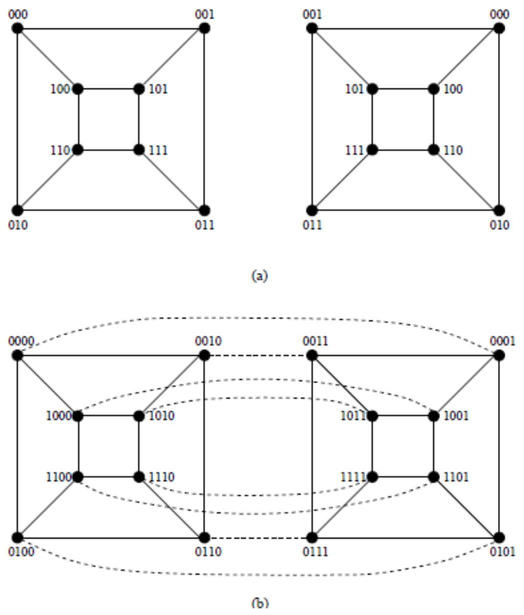


Figure 6 Hypercubes H_3 and H_4

In hypercube $\mathrm{H_m}$ all vertices have degree m, the diameter is m and there are $\mathrm{m2^{m-1}}$ edges. The broadcast time of hypercube is m, which is easily justified by noticing that at each round all informed vertices inform an uninformed neighbor, such that at each round the number of informed vertices doubles.

The Cube-Connected Cycles CCC_m is the graph obtained by replacing each vertex of the hypercube H_m with a cycle of m vertices. The i-th dimension edge incident to a node of the hypercube is then connected to the i-th node of the corresponding cycle of the CCC_m. Figure 7 shows a 3-dimensional Cube-Connected Cycles graph.

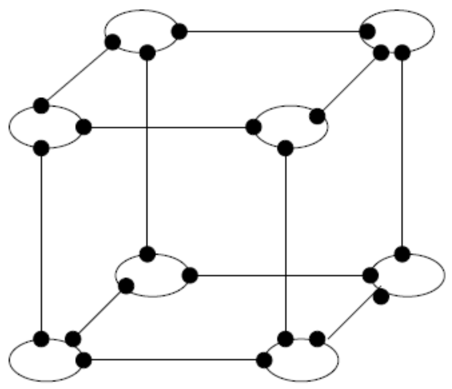


Figure 7 Cube-Connected Cycles CCC_3

A cube-connected cycles CCC_m has $m2^m$ vertices, each vertex has degree 3, and the diameter is equal to $2m + \left\lfloor \frac{m}{2} \right\rfloor - 2$. The optimal broadcast scheme is that every informed vertex informs first its hypercube neighbor, and then informs the right neighbor on the cycle and in the end the left neighbor. The broadcast time of this scheme is $b(CCC_m) = \left\lceil \frac{5m}{2} \right\rceil - 1$ as shown in [30].

The Shuffle-Exchange $\mathrm{SE_m}$ is a graph whose vertices are represented by binary strings of length m. Figure 8 shows a Shuffle-Exchange graph of dimension 3.

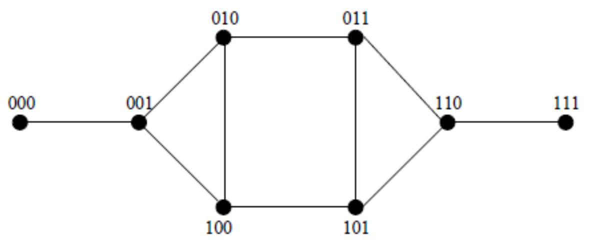


Figure 8 Shuffle-Exchange SE_3

Each vertex αa of SE_m, where α is a binary string of length $m-1$ and a is in $\{0,1\}$, is connected to vertices $a\alpha$ and αc, where c is the binary complement of a. The Shuffle-Exchange is a graph with 2^m vertices, maximum degree 3, and diameter $2m-1$. The exact broadcast time is not known, but an upper bound $b(SE_m) \leq 2m-1$ was proved by [23].

The DeBruijn DB_m is a graph whose vertices can be represented by binary strings of length m and each vertex $a\alpha$, where α is a binary string of length $m-1$ and a is in $\{0,1\}$, is connected to vertices αb, where b is in $\{0,1\}$. Figure 9 shows a DeBruijn graph of dimension 3.

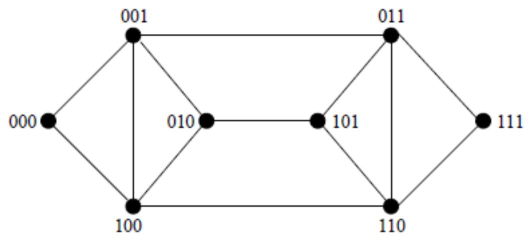


Figure 9 DeBrujin DB_3

DB_m has 2^m vertices, diameter m and maximum degree 4. The lower bound of the broadcasting time is $b(DB_m) \geq 1.3171m$, proven in [27]. The upper bound is $b(DB_m) \leq 1.5m + 1.5$, proven in [6].

The Butterfly BF_m is a graph whose vertices are in $\mathrm{V_m} = \{0,1,\dots,m-1\} \times \{0,1\}^m$, where $\{0,1\}^m$ denotes the set of length-m binary strings. For each $i \in \{0,1,\dots,m-1\}$, $\alpha \in \{0,1\}^m$ the vertex (i,α) is connected to vertices $\big((i+1) \bmod m, \alpha\big)$ and $\big((i+$

$1) \bmod m, \alpha(i))$, where $\alpha(i) = a_0 \dots a_{i-1} c_i a_{i+1} \dots a_{m-1}$, and c_i is the binary complement of a_i. Figure 10 shows a Butterfly graph of dimension 3.

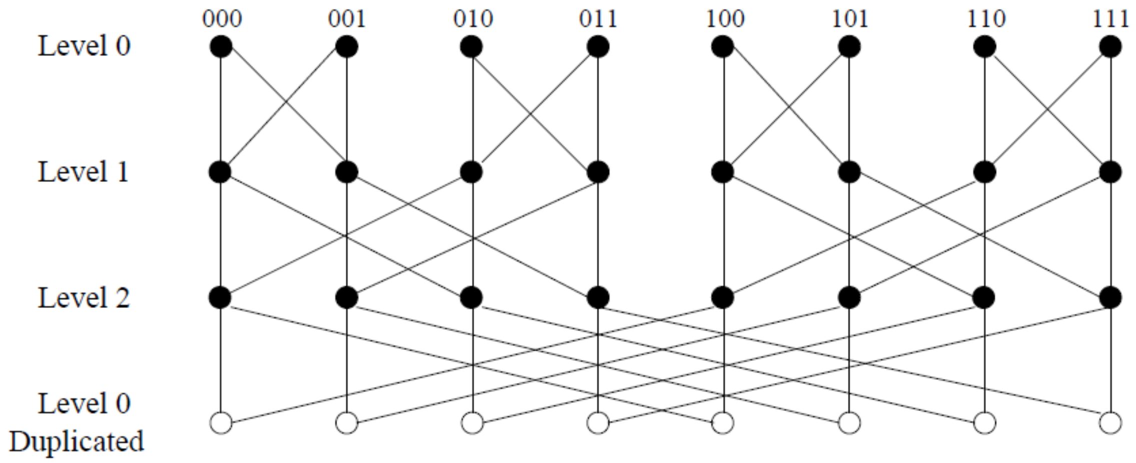


Figure 10 Butterfly BF_3

The number of vertices of $\mathrm{BF_m}$ is $\mathrm{m2^m}$, the maximum degree is 4 and the diameter of $\mathrm{BF_m}$ is $\left\lfloor \frac{3m}{2} \right\rfloor$. The broadcast time of an m-dimensional Butterfly graph is $1.7417m \leq b(BF_m) \leq 2m - 1$, proven in [27].

The d-Grid $G[a_1 \times a_2 \times \dots \times a_d]$ is a graph whose vertices are represented by d-tuples of positive integers $(z_1, z_2, \dots, z_d)$, where $0 \leq z_i < a_i$ for all $i \in \{1,2,\dots,d\}$. Each edge connects two vertices whose d-tuples differ in exactly one coordinate by exactly one. For example, in $G[4 \times 3]$, shown in Figure 11, vertex $(1,1)$ is connected to vertices $(0,1)$, $(2,1)$, $(1,0)$ and $(1,2)$.

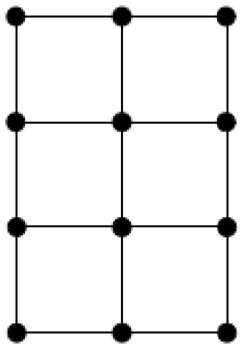


Figure 11 d-Grid $G[4 \times 3]$

The number of vertices of $G[a_1 \times a_2 \times \ldots \times a_d]$ is $a_1 \times a_2 \times \ldots \times a_d$, the maximum degree is $2d$ when $a_i \geq 3$, and the diameter is $(a_1 - 1) + (a_2 - 1) + \cdots + (a_d - 1)$. The broadcast time of a 2-Grid $G[a_1 \times a_2]$ is $a_1 + a_2 - 2$ as presented in [20].

The $\boldsymbol{d}$-Torus $T[a_1 \times a_2 \times \ldots \times a_d]$ is a graph created by connecting both ends of each dimension of a d-Grid graph. Figure 12 shows a d-Torus graph of dimension 4×3.

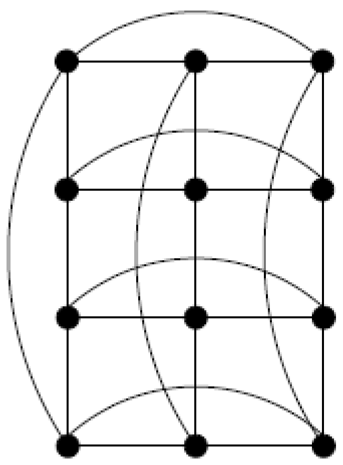


Figure 12 d-Torus $T[4 \times 3]$

The broadcast time of a 2-Torus $T[a_1 \times a_2]$ is $\left\lceil\frac{a_1}{2}\right\rceil + \left\lceil\frac{a_2}{2}\right\rceil$, when $a_1$or a_2 is even, and it is $\left\lceil\frac{a_1}{2}\right\rceil + \left\lceil\frac{a_2}{2}\right\rceil - 1$, when both $a_1$and a_2 are odd [13].

The Knödel graph $W_{\Delta,n}$ is a graph on $n \geq 2$ vertices (n even) and of maximum degree $\Delta \geq 1$. The vertices of $W_{\Delta,n}$ are represented as (i,j) pairs, where $i = 1,2$ and $0 \leq j \leq (n/2) - 1$. For every j, $0 \leq j \leq (n/2) - 1$, there is an edge between vertex $(1,j)$ and every vertex $(2, j + 2^k - 1 \; mod \; n/2)$, for $k = 0, \dots, \Delta - 1$ [16].

For $0 \leq k \leq \Delta - 1$, an edge that connects a vertex (i,j) to the vertex $(2, j + 2^k - 1 \; mod \; n/2)$ is said to be in dimension k. Figure 13 shows two Knödel graphs, with $\Delta = 3$ and $n = 14$ and $n = 8$, respectively.

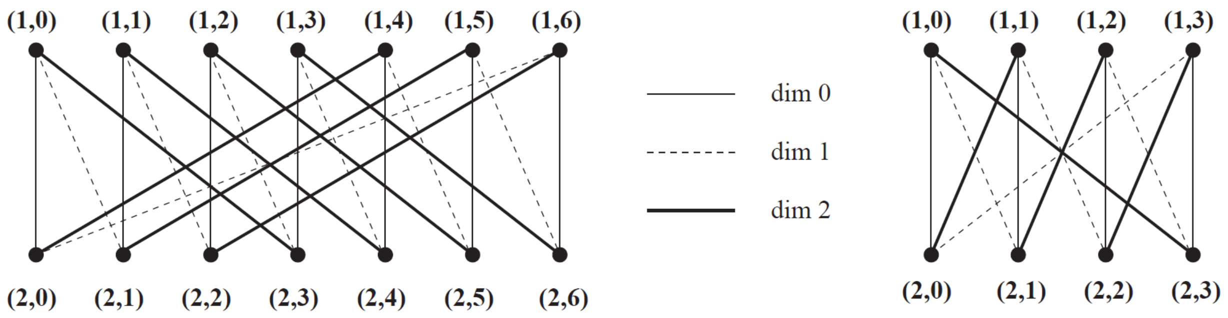


Figure 13 $W_{3,14}$ and $W_{3,8}$

2.2 Existing Approximation and Heuristic Algorithms

The minimum broadcast time problem in arbitrary networks has been examined since the late 1970s [14]. In 1981, Scheuermann et al. publish their first attempts to solve the problem [34]. Then, in 1984, Scheuermann et al. present an exact solution based on dynamic programming to the problem of optimal broadcasting in arbitrary networks, but this exact algorithm is not computationally efficient in large networks [35].

As described in the NP-Completeness section, the problem of finding the minimum broadcast time of an arbitrary originator in an arbitrary graph is NP-complete [37]. As a result, a large number of approximation algorithms and heuristic algorithms have been proposed. This section examines some of the most important ones.

2.2.1 Approximation Algorithms

An approximation algorithm is an algorithm that does not provide an optimal solution but which, nevertheless, provides a solution that is proven to be within a certain degree of proximity to the optimal solution.

Kortsarz and Peleg [28] present one of the first approximation algorithms for broadcasting in arbitrary graphs. For an arbitrary graph $G = (V, E)$, their algorithm is an $\mathrm{O}\left(\sqrt{|V|}\right)$-additive approximation algorithm. Ravi introduces an algorithm for the broadcasting problem which gives an $O\left(\frac{log^2|V|}{log\, log|V|}\right)$-approximation [33].

There are also several approximation algorithms that attempt to improve broadcasting time using methods that are not focused solely on the optimization of the bound time. Hoelting et al. propose a genetic algorithm that generates a heuristic of complexity $O(mn^3)$ by using a global precedence vector [21]. An integer programming formulation is used in [2] to generate an $O(\log n)$ approximation algorithm. Finally, Barth et al. propose a general approach to structured communication that can be applied to any network to improve the broadcasting time therein [4].

To the best of our knowledge, the optimal theoretical upper bound approximation is presented in [11], a broadcast approximation of $\mathrm{O}\left(\frac{\log|\mathrm{V}|}{\log\log|\mathrm{V}|}\mathrm{b}(\mathrm{G})\right)$ rounds.

In addition to being NP-complete, the problem of finding a minimum broadcast time of an arbitrary originator in an arbitrary graph is also difficult to approximate [2, 12, 33]. As a result, one must go beyond approximation algorithms and consider heuristic algorithms to attempt to resolve the minimum broadcast time problem.

2.2.2 Heuristic Algorithms

A heuristic algorithm is an algorithm that solves a hard problem, by providing a solution that is not optimal, complete or accurate, but it has good performance results in practice and it is fast to find and implement.

The following subsections describe several existing heuristic algorithms used to solve the minimum broadcasting time problem. These heuristics will be used as the baseline for comparison with the new heuristic introduced in the section Proposed Heuristic Algorithm.

The performance of existing heuristic algorithms varies based on the topology of the network in which they are applied. For instance, Round Heuristic [5] and the Tree Based Algorithm [18] offer good performance in most commonly used topologies, and even better performance in network models from ns-2 simulator [1, 2, 10, 42]. The Random and Semi-Random heuristics [38] have, in general, worse performance than Round Heuristic (RH) and the Tree Based Algorithm (TBA) in commonly used topologies, but in ns-2 simulator network models they generate better results in graphs of GT-ITM Transit-Stub model and Tiers model.

2.2.2.1 Round Heuristic

The Round Heuristic [5] assigns a weight to each edge of a network, activates matched edges using a maximum weighted matching algorithm, and uses the active edges to transfer the message through the network. This procedure is performed during each broadcasting round and is continued until the message is transmitted throughout the entire network.

The simulation results in a number of graphs [5] show that the performance of the Round Heuristic comes close to the minimum broadcasting time in a graph.

Logical and efficient assigning of weights to each edge of a graph is a fundamental part of the Round Heuristic. Two different approaches can be used to perform the assignment: Potential Approach and Breadth-First-Search (BFS).

In the Potential Approach, a weight of 0 or 1 is assigned to each edge (v, w), where 1 is assigned if either v or w know the message to broadcast and 0 otherwise.

Although it is fast and does not require large amounts of memory, the Potential Approach is somewhat simplistic, focusing solely on the current location in the network, and lacking a global view. Therefore, typically this approach is not used.

Although it is generally slower and uses more memory than the Potential Approach, the Breadth-First Search is much more comprehensive, and is the approach typically used. Several important concepts must be defined prior to describing the BFS approach.

The dispersion region $DR(p, t)$ of a message p in a connected graph is the set of vertices that know message p at the beginning of round t. For any vertex v, we denote $dist_v(p, t)$ the shortest distance in the graph from vertex v to a vertex $w \in DR(p, t)$. **The set of border-crossing edges** $bce(p, t)$ is defined as $bce(p, t) = \{(v, w) \in E | v \in DR(p, t) \; and \; w \notin DR(p, t)\}$. We also denote by $bce_v(p, t)$ the subset of all edges in

$bce(p,t)$ that lie on the shortest path from $DR(p,t)$ to v, for any vertex $v \in DR(p,t)$. Figure 14, adapted from [5], illustrates the dispersion region $DR(p,t)$ for a message p at round t. The border-crossing edges are shown in bold. $dist_v(p,t) = 3$ and $bce_v(p,t) = \{e_1, e_2\}$.

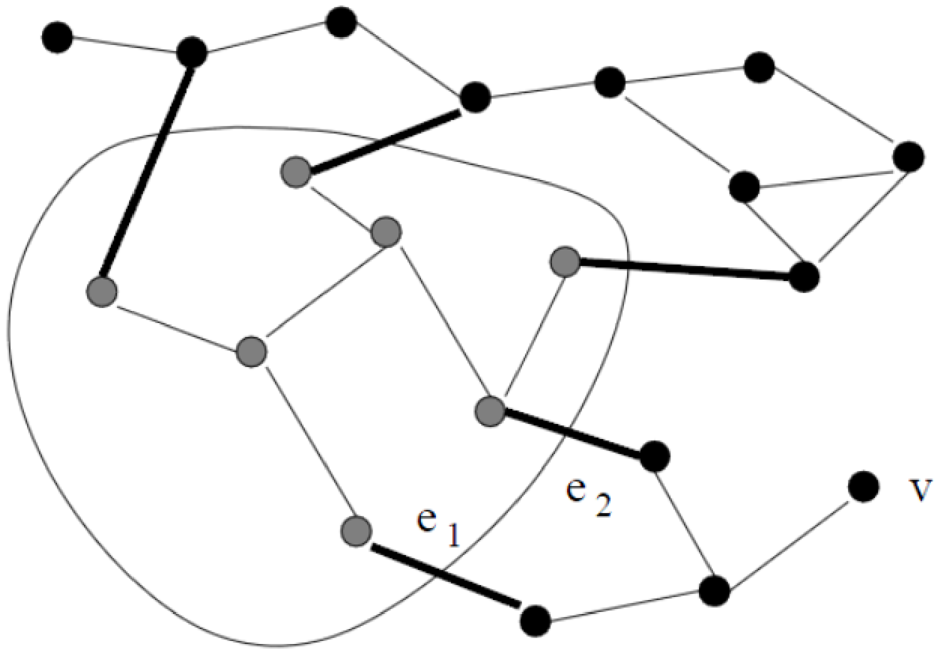


Figure 14 The dispersion region $DR(p,t)$ for message p at round t

At each round, border-crossing edges can be used to spread p further in that round. Each border-crossing edge will be assigned a weight that represents the sum of the contributions by each message p. The weight indicates how useful a given edge $e \in bce(p,t)$ is for the broadcasting of message p, and it is calculated taking into account how useful e is and how far v is from $DR(p,t)$. The best route for message p is through a shortest path from $DR(p,t)$ to all uninformed vertices. Edge $e \in bce(p,t)$ is considered more useful, if it lies on many of these shortest paths. Another criteria considered to calculate the weight of e is $dist_v(p,t)$. The further away v is from $DR(p,t)$, the higher the priority to forwarding p towards v, hence the higher weight of e. Based on the above, the weight that all vertices $v \in DR(p,t)$ contribute to every edge $e \in bce_v(p,t)$ is calculated using the formula below:

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

$Dist\_Exp$ and Num_Exp are two parameters and for every vertex v, at round t, $dist_v(p,t)$ and $bce_v(p,t)$ are calculated. A modified breadth first search algorithm [5] is used to calculate $dist_v(p,t)$ and $bce_v(p,t)$ in order of increasing $dist_v(p,t)$. When $dist_v(p,t) = 1$, then $bce_v(p,t)$ contains all edges that connect v to a vertex in $DR(p,t)$. When $dist_v(p,t) > 1$, then $bce_v(p,t)$ is the union of the sets $bce_{w_i}(p,t)$, for all vertices w_i adjacent to v with $dist_{w_i}(p,t) = dist_v(p,t) - 1$.

Calculating $bce_v(p,t)$ for any vertex v requires analyzing at most $|V|$ vertices with at most $|E|$ edges each, taking $O(|V||E|)$ time. For all vertices v this calculation takes $O(|V|^2|E|)$, which is also the time it takes to calculate the weights for one round. So, without taking into account the matching step, the Round Heuristic takes $O(R|V|^2|E|)$, where R is the number of rounds of broadcasting.

The two parameters $Dist\_Exp$ and Num_Exp have a great impact on the value of the weight, thus playing a significant role in the performance of the heuristic. The simulation results of the Round Heuristic in commonly used topologies give in some cases results close or equal to the optimal broadcast time [5]. We will look closer at these results in Chapter 4.

2.2.2.2 Tree Based Algorithm

The Tree Based Algorithm (TBA) presented in [18] builds upon the ideas from the Round Heuristic. In round t, TBA separates the graph into two regions, the bright region and the dark region. The bright region consists of all informed vertices, similar to the Dispersion Region in Round Heuristic, while the dark region consists of all uninformed vertices. All informed vertices that have neighbors in the dark region are called the bright border, denoted by $bb(t)$. Given an uninformed vertex u and its uninformed neighbor v, we say u is a child of v, if $D(u,t) = D(v,t) + 1$, where $D(v,t)$ stands for the shortest distance from uninformed vertex v to $bb(t)$. The children and the children's children are all called the descendants.

Figure 15, adapted from [18], shows the definitions in TBA. Vertex a is the originator. After three rounds, all vertices in the dark region are still uninformed. The informed vertices, d, g, f with shadowed backgrounds belong to $bb(4)$. Vertices o and p are children of vertex j, and vertex q is a child of vertices o and p. In other words, vertices o, p and q are descendants of vertex j.

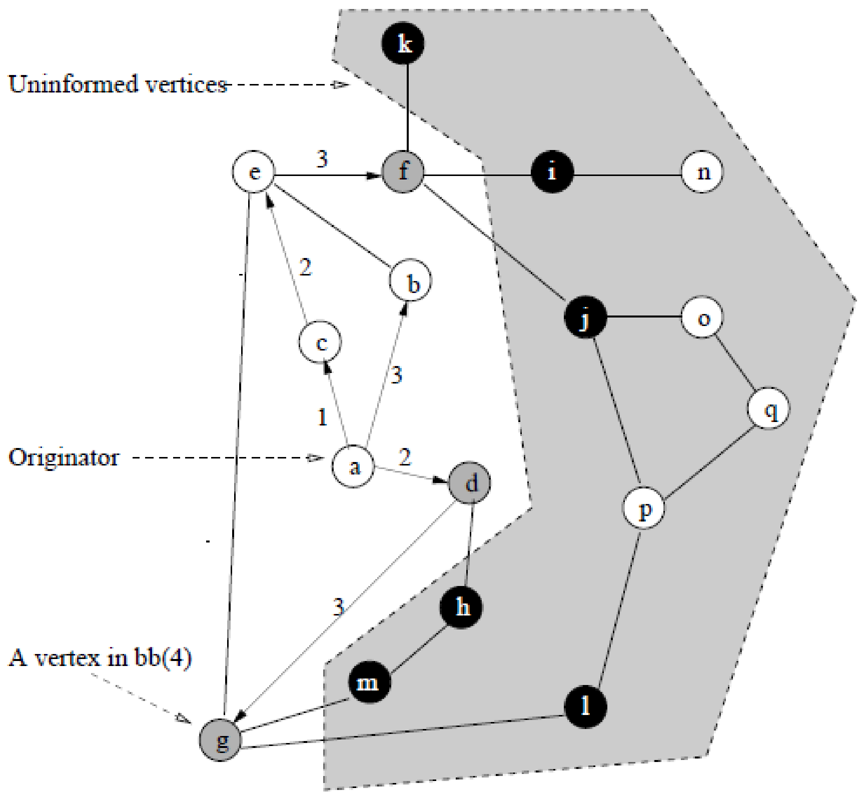


Figure 15 Definitions in TBA.

At each round, the TBA computes a matching between the set of informed and uninformed vertices and then disseminates the message to the uninformed vertices using this matching. A modified breadth first search algorithm (BFS), from the vertices on the bright border $bb(t)$ towards the uninformed vertices, is used to compute the matching. The modified BFS algorithm labels all uninformed vertices v with $D(v,t)$. As described above, the distances $D(v,t)$ define the parent-child relationships between the uninformed vertices. Then, for each uninformed vertex u, its weight at round t, denoted by $w(u,t)$, is calculated using the strategy of the optimal broadcasting in trees. If u has no children, then $w(u,t) = 0$. Otherwise, if u has p children $v_1, v_2, \dots, v_p$, arranged in decreasing order of their weights, $w(v_1, t) \geq w(v_2, t) \geq \cdots \geq w(v_p, t)$, then the weight assigned to vertex u at round t is $w(u, t) = \max_{1 \leq i \leq p}\{w(v_i, t) + i\}$. In the next step, a maximum weighted matching is computed between uninformed and informed vertices using a heuristic that aims firstly to maximize the number of pairs of vertices in the matching and secondly to maximize the weights of matched vertices. In the last step, each matched informed vertex will inform its uninformed pair.

The number of rounds needed to inform all vertices is the broadcast time. From [18] we learn that the time complexity of each round is $O(|V| + |E|) = O(|E|)$, resulting in a total time complexity of $O(R|E|)$.

A refined version of the TBA calculates the weight of a child depending on the number of parents, allowing TBA to obtain better results in some topologies. The refined version has the same time complexity as the original heuristic.

Simulation results for commonly used topologies and other network models show that TBA has good results in practice, even better than the Round Heuristic in most cases.

2.2.2.3 Minimum-Weight Cover Heuristic

The Minimum-Weight Cover (MWC) heuristic applies the MWC algorithm from [28] in the layer graph of a connected graph $G = (V, E)$. To better understand the heuristic, we will first define the layer graph and the MWC problem, subsequently combining them together to solve the broadcasting problem.

Definition 1 *Given an originator o and any vertex v, the layer of v, denoted by $L(v)$, is the shortest distance from o to v. A graph $G_L = (V_L, E_L)$ is called a layer graph of graph G, where $V_L = V$ and for any edge $(u,v) \in E$, $(u,v) \in E_L$ iff $L(v) = L(u) + 1$.*[38]

Figure 16 shows an example of a layer graph where it is obvious that each two adjacent layers of the layer graph constitute a bipartite graph.

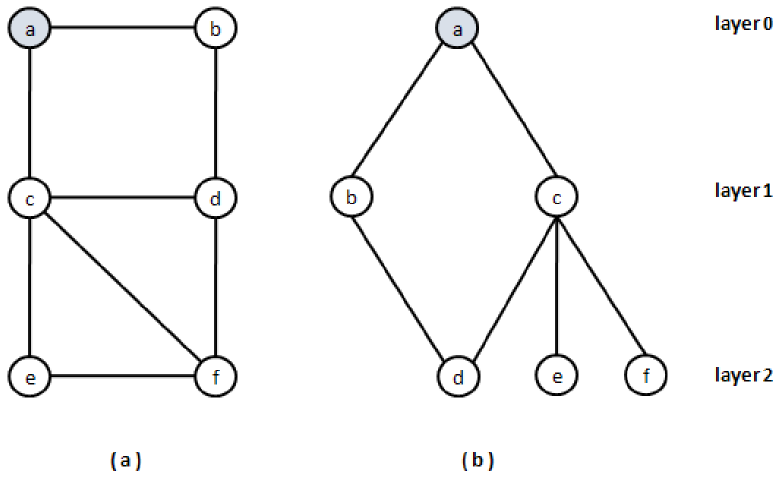


Figure 16 (a) The original graph G. (b) The layer graph G_L

The Minimum-Weight Cover problem presented in [28] is stated below.

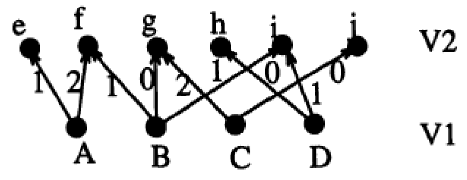
Let $G(V_1, V_2, A, w)$ be a bipartite graph with bipartition (V_1, V_2), edge set A, and a weight function $w : A \rightarrow Z^+$ on the edges, and no isolated vertices. A control function $F : V_2 \rightarrow V_1$, where $F(v_2) = v_1$ implies that $(v_1, v_2) \in A$, and we say that v_1 controls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

For every server $v \in V_1$, denote the customers dominated by v by $D_1(v), \dots, D_k(v)$, and denote the edges connecting v with its customers by $e_i^v = \big(v, D_i(v)\big)$. Assuming, without loss of generality, that all the customers dominated by v are ordered such that $w(e_i^v) \geq w(e_{i+1}^v)$ for every i, the weight of F is defined as

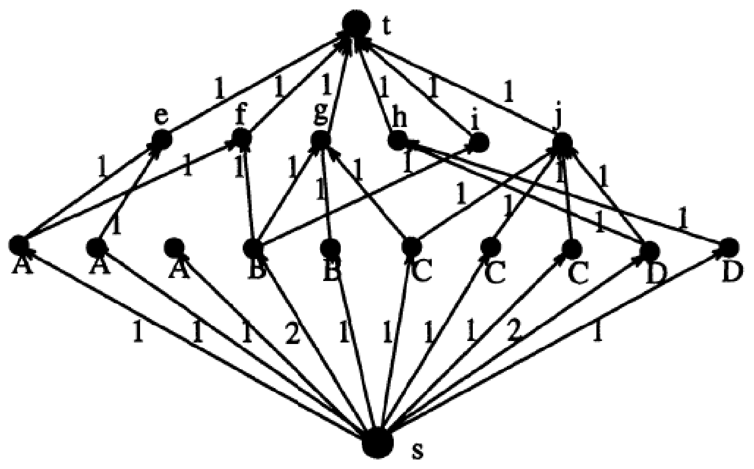
$$W(F) = \max_{v \in V_1} \left\{ \max_i \{ i + w(e_i^v) \} \right\}$$

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 $W(F)$ is minimal. The function F is called the *minimum control function* for G.

The problem is solved using a pseudo polynomial algorithm. The algorithm verifies whether there exists a positive integer j, and a control function F, such that $W(F) \leq j$, where $\min_e\{w(e)\} + 1 \leq j \leq \min_e\{w(e)\} + |V_2|$. Starting from G, the algorithm constructs a modified flow graph G'_j. In [28] it is proved that G has a control function F with weight $W(F) \leq j$, iff it is possible to push $|V_2|$ units of flow from the source to the sink on G'_j.



(a) Bipartite Graph



(b) The flow graph

Figure 17 (a) A bipartite graph G. (b) Its corresponding flow graph G'_3.

The construction of the flow graph G'_j starts by adding a source vertex s and a sink vertex t to the original graph G. Then, assuming that w_v is the maximal weight that is less than or equal to $j-1$ of an edge incident to $v \in V_1$, duplicate v into $w_v + 1$ 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 $j - w_v$ 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 $w(v, u) \leq w_v - i + 1$. Finally, for each customer $u \in V_2$, create a directed edge (u, t) with capacity 1.

The MWC algorithm is given below.

```
1. Start with $j_1 = \min_e\{w(e)\} + 1$ and $j_2 = \max_e\{w(e)\} + |V_2|$.
2. Repeat
    a. $j = \left\lceil \frac{j_1 + j_2}{2} \right\rceil$.
    b. Construct the flow graph $G'_j$.
    c. Compute the maximal flow on $G'_j$ from source to sink.
    d. If the maximal flow is $|V_2|$
        i. Then set $j_1 = j$
        ii. Else set $j_2 = j$
3. Until $j = j_2 \leq j_1 + 1$.
4. Return the minimum control function $F$ corresponding to the maximal flow
   computed on $G'_{j_1}$.
```

To solve the minimum time broadcasting problem, MWC heuristic first constructs a layer graph G_L by performing a breadth-first search starting from the originator o in the arbitrary graph G and removing all the edges that have not been traversed during the breadth-first search. Then, it performs the MWC algorithm between adjacent layers in the layer graph as well. A spanning tree will be generated, and based on that we can obtain the broadcast scheme.

Dinic's maximum flow algorithm has complexity $O(|V|^2|E|)$. When applying Dinic's maximum flow algorithm to the MWC algorithm, the total time complexity of the MWC algorithm is $O(|V|^2|E| \log|V|)$.

2.2.2.4 Minimum-Weight Cover Modified Heuristic

In the previous section we looked at an algorithm that, for any bipartite graph Let $\mathrm{G}(\mathrm{V}_1, \mathrm{V}_2, \mathrm{A}, \mathrm{w})$ with bipartition $(\mathrm{V}_1, \mathrm{V}_2)$, edge set A, and a weight function w, finds the minimum value of $max_{v \in V_1}\{max_i\{i + w(e_i^v)\}\}$. A modified MWC algorithm which aims to minimize $max_{v_j \in V_1}\{max_i\{i + w(e_i^v)\} + j\}$ is proposed in [38].

Starting from the control function $F : V_2 \rightarrow V_1$ generated by the MWC algorithm the MWC-Modified algorithm first labels all servers with a weight $w_{S_j} = max_{e_i^{v_j} \in F}\{i + w\left(e_i^{v_j}\right)\}$. Then, the servers are ordered in descending order of their weights $w_{S_1} \geq w_{S_2} \geq \cdots \geq w_{S_{|V_1|}}$. Finally, for each MW going from $w_{S_1} + 1$ to $max_F\left\{w_{S_j} + j\right\}$, generate a new control function F' such that $max_{F'}\left\{w_{S_j} + j\right\} \leq MW$, keeping MW as small as possible.

The MWC-Modified algorithm does not increase the complexity of the MWC algorithm. The additional step that analyzes all servers, trying to reduce their weights by modifying the control function, has a complexity of $O(|V|^2|E|)$. Therefore the total complexity of the MWC-Modified is still $O(|V|^2|E|\, log|V|)$ when using Dinic's maximum flow algorithm for the MWC part.

2.2.2.5 Random Heuristic

The Random Heuristic uses the shortest paths to disseminate the message from the originator to the rest of the graph. Similar to the MWC Heuristic, first, a layer graph is constructed from the original graph. A parent-child relationship is defined between the neighbor vertices in the layer graph as follows: v is a child of u if u and v are connected and $L(v) = L(u) + 1$. Implicitly, u is a parent of v. The children of vertex u are its descendants, and the children of the descendants of u are also descendants of u. The heuristic uses the concept of estimated broadcast time of a vertex v in graph G, denoted by $EB(v)$, and calculated using the following recursion.

1. $EB(v)$ is equal to 0, if vertex v has no children.
2. If v has k children, $c_1, c_2, \dots, c_k$, and all these children are in the descending order of $EB(c_i)$, i.e., $EB(c_i) \geq EB(c_{i+1})$, then $EB(v) = \max \{EB(c_i) + i\}$, where $1 \leq i \leq k$.

In [19], a linear algorithm is presented to calculate $EB(v)$ if the estimated broadcast times of the children of v, $EB(c_i)$, are given, where c_i is a child of vertex v, and $1 \leq i \leq k$. The complexity of this algorithm is $O(k)$.

Furthermore, in [19] it is also proved that, for any vertex v in a tree, $\mathrm{EB(v)}$ is exactly the time that vertex v broadcasts the message to all of its descendants.

The Random Heuristic is very simple and has the three steps below.

1. Construct the layer graph $G_L$of the arbitrary graph G by performing a breadth-first search starting from the originator o and removing all the edges that have not been traversed during the breadth-first search.

2. Construct a spanning tree as follows. For every two adjacent layers of G_L, randomly match children with only one of their parents and remove unused edges between the two adjacent layers.
3. Calculate $\mathrm{EB(v)}$ for each vertex v of the spanning tree.

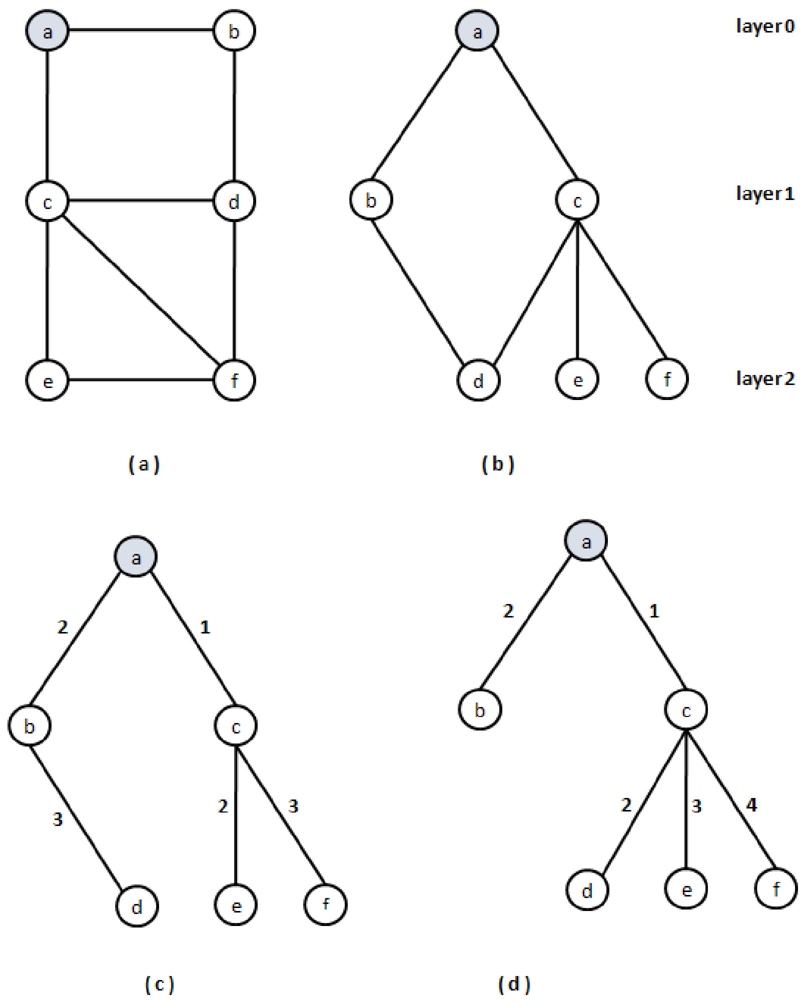


Figure 18 (a) The original graph G with originator a. (b) The layer graph G_L. (c) and (d) Possible broadcast schemes with different broadcast times, 3 and 4 respectively.

The broadcast time of o in graph G is $\mathrm{EB}(o)$.

The complexity of the algorithm is the sum of the complexities of the three steps. Constructing the layer graph takes $O(|E|)$ time, constructing the spanning tree takes $O(|V|)$ time since all vertices are randomly assigned a parent, and finally, calculating $EB(v)$ for all vertices v takes $O(|E|)$ time. The total time complexity of the Random Heuristic is $O(|E|) + O(|V|) = O(|E|)$.

2.2.2.6 Semi-Random Heuristic

The Random Heuristic is very simple and has low complexity, but it has random steps, therefore it could potentially be improved. The Semi-Random heuristic, described in [19], replaces the random matching of children to a parent by a strategy that aims to minimize $\max \{\mathrm{EB}(\mathrm{p_i})\}$, for each parent $\mathrm{p_i}$ on the same layer.

The Semi-Random heuristic presented in [19] is as follows.

1. Construct the layer graph $G_L$of the arbitrary graph G by performing a breadth-first search starting from the originator o and removing all the edges that have not been traversed during the breadth-first search.
2. Assuming that G_L has k layers, label all vertices v_i on layer $k-1$ with $EB(v_i)$.
3. For each layer l starting from $k-2$ to 1, call procedure SRM.
4. The broadcast time of o in graph G is $EB(o)$.

The procedure SRM receives as input all vertices on layers l and $l+1$ and is as follows.

1. On layer l, for each parent p_i do the following:

   a. Match p_i with those children that have different weights and remove all edges that connect these children with other parents;
   b. Label p_i with $EB(p_i)$ not including those unmatched children.
2. For each unmatched child, do the following:
   a. Match it with the parent of the smallest weight, and remove its edges connecting to other parents;
   b. Update the weight $EB(p)$ for its matched parent p.

The Semi-Random heuristic has the same time complexity as the Random heuristic. The complexity of the construction of the layer graph and the complexity of the procedure SRM are both $O(|E|)$. Their sum makes the total complexity of the Semi-Random heuristic equal to $O(|E|)$.

3 Proposed Heuristic Algorithm

In this chapter we will propose a new heuristic algorithm, which aims to improve the existing heuristics. We have seen in the previous chapter that the Semi-Random heuristic has a low complexity and good results in practice, but it still makes potentially non optimal decisions by randomly matching children with parents. The new heuristic proposes a new strategy to match children and parents from adjacent layers of the layer graph, aiming to make less random decisions.

We recall the concept of estimated broadcast time of a vertex v in graph G, denoted by $EB(v)$, which we have already seen above in [19]. In this chapter, the term weight of v is interchangeably used to refer to the estimated broadcast time of v. The following recursion is used to calculate $EB(v)$.

1. $EB(v)$ is equal to 0, if vertex v has no children.
2. If v has k children, $c_1, c_2, \dots, c_k$, and all these children are in the descending order of their weights $EB(c_i)$, i.e., $EB(c_i) \geq EB(c_{i+1})$, then $EB(v) = \max \ \{EB(c_i) + i\}$, where $1 \leq i \leq k$.

The new strategy to match children and parents from adjacent layers of the layer graph attempts to minimize the $max \ \{EB(p_i)\}$, for all parents p_i on the same layer.

Additionally, we have observed that in sparse graphs, removing the edges that are not needed while constructing the layer graph has a significant negative impact on the overall performance of the algorithm. Therefore, our new heuristic does not remove these edges completely; instead it keeps them as inactive and tries to take advantage of them once the layer by layer matching of children to parents is completed.

3.1 Algorithm

The new heuristic algorithm is presented below.

1. Construct the layer graph G_L of the arbitrary graph G by performing a breadth-first search starting from the originator o and marking as inactive all the edges that have not been traversed during the breadth-first search.
2. Assuming that G_L has k layers, numbered from 0 to $k-1$, label all the vertices v_i of layer $k-1$ with $EB(v_i)$.
3. For each layer l starting from $k-2$ to 1, call procedure *Matching* to match children to one parent.
4. Perform procedure *Broadcast and Improve* on the resulting spanning tree.
5. The broadcast time of o in graph G is the number of rounds returned by procedure *Broadcast and Improve*.

PROCEDURE *Matching*

Input: All vertices on layer l and layer $l+1$ and all edges between these two layers in the layer graph.

1. Order the parents by decreasing number of children and denote the parents by $p_1, p_2, \dots, p_n$ where p_1 has the most number of children and p_n the least number of children.

2. Start with an empty matching.
3. Add p_1 and all its children to the matching.
4. Match p_1 with all its children.
5. For each parent p_k, starting from p_2 to p_n, take the following actions:
    a. Add p_k and all its children to the matching.
    b. Match p_k with all its unmatched children at this step (those who have p_k as single parent)
    c. Reposition p_k in the matching such that the parents in the matching are ordered by decreasing estimated broadcast time $EB(p_i)$.
    d. Find the parent p_i which gives $b_{max} = max_{1 \le i \le k}\{i + EB(p_i)\}$ and denote this parent as p_{max}.
    e. If p_{max} is p_k then remove all unused edges and go to step 5.
    f. If the number of children common to both p_{max} and p_k is 0, then remove all unused edges and go to step 5.
    g. If $b_{max} = b(p_k)$ then remove all unused edges and go to step 5.
    h. From the children common to both p_{max} and p_k, choose the maximum weight child which has the same weight as another child of p_{max}. Remove the edge matching it to p_{max} and match it to p_k.
    i. Go to step 5.c.

PROCEDURE *Broadcast and Improve*

Input: A spanning tree, including inactive edges between siblings and the originator o.

Output: A broadcast scheme in the spanning tree and a number of rounds required to broadcast a message in the spanning tree.

```
1. Inform originator $o$ and order its children in decreasing order of their $EB$.
2. While there remain uninformed vertices do
    a. For each informed vertex $v$ do
        i. If $v$ has uninformed children, inform its next uninformed child and
           order the child's children in decreasing order of their $EB$.
        ii. Else if $v$ has uninformed siblings, inform the next uninformed
            sibling that has maximum $EB$ and order the children of that
            sibling by decreasing order of their $EB$.
3. Output the broadcast scheme and the number of times the step 2 was
   performed.
```

The procedure Matching is the most important part of the algorithm. Its main goal is to assign children to parents in such a way that $max\ \{EB(p_i)\}$ becomes as small as possible. To achieve this, at each step, the algorithm tries to remove children from the parent with maximum EB at that point and assign to the parent processed at that step. For a better understanding of the algorithm, an example of the procedure *Matching* is presented below. Figure 19 shows a bipartite graph between two layers of the layer graph G_L representing the input to procedure *Matching*. The estimated broadcast time for each child is labeled on the graph.

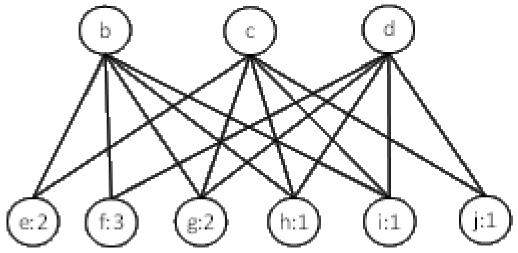


Figure 19 The original bipartite graph between two layers of the layer graph $\mathrm{G_L}$

After ordering the parents by decreasing number of children, parent b with the most children is added to the matching and its estimated broadcast time is calculated. Figure 20 shows the matching at this point.

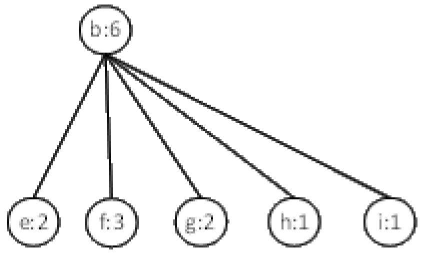


Figure 20 Parent b is added to the matching

Next step is shown in Figure 21. Parent d, the next one in order of decreasing number of children, is added to the matching together with all its children that have not already been added before. In this case child j is the only child of d that has not already been added, so j gets added to the matching and matched with d. The dashed lines represent unused edges that d adds to the matching and which could potentially be used in the next steps to minimize the estimated broadcast time of b.

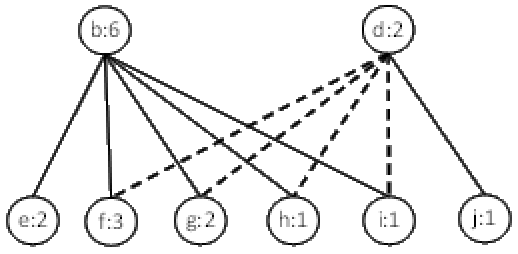


Figure 21 Parent d is added to the matching

At this point, d will try to minimize the estimated broadcast time of b by taking over one of b's children. Since the children with the same broadcast time are the ones that increase the estimated broadcast time of their parent, the algorithm chooses from the children with the same broadcast times, a child that has the max broadcast time among the children that are common to b and d. In Figure 22 we see that g is moved from b to d, hence the broadcast time of b decreases to 5 and the broadcast time of d increases to 3.

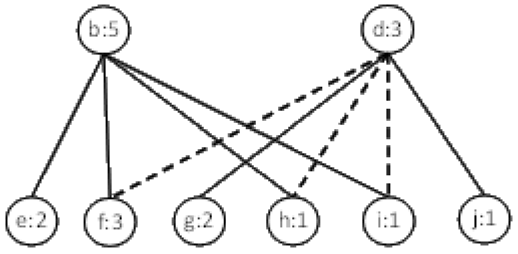


Figure 22 Child g is moved from parent b to d

Since the broadcast time of d is still less than the broadcast time of b, the previous step is repeated and child i gets moved from b to d, making the broadcast times of b and d equal to 4.

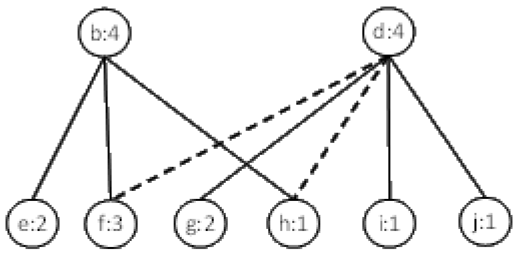


Figure 23 Child i is moved from parent b to parent d

Because the broadcast time of the current parent d is now equal to the broadcast time of b, the algorithm considers the processing of parent d complete and moves to the next parent. Parent c and all its children and edges are added to the matching. Since c does not have any children of its own, its initial broadcast time is 0.

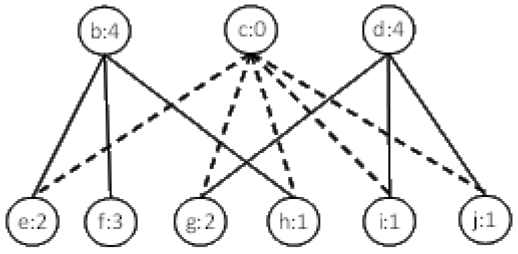


Figure 24 Parent c is added to the matching

At this point, in decreasing order of their estimated broadcast time $EB(p_i)$, parents b, d, c respectively have:

- $b(b) = i_b + EB(b) = 1 + 4 = 5$
- $b(d) = i_d + EB(d) = 2 + 4 = 6$
- $b(c) = i_c + EB(c) = 3 + 0 = 3$

hence $b_{max} = \max_{1 \le i \le k}\{i + EB(p_i)\}$ is given by parent d, with $b(d) = 6$, therefore parent c takes over child j from parent d and the resulting matching is shown in Figure 25.

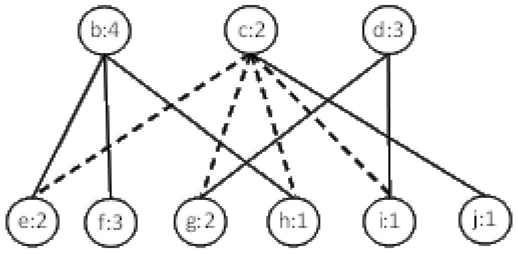


Figure 25 Child j is moved from parent d to parent c

Once again, in decreasing order of their estimated broadcast time $EB(p_i)$, parents b, d, c respectively have:

- $b(b) = i_b + EB(b) = 1 + 4 = 5$
- $b(d) = i_d + EB(d) = 2 + 3 = 5$
- $b(c) = i_c + EB(c) = 3 + 2 = 5$

Since c has now become one of the parents whose $b(c) = b_{max} = max_{1 \le i \le k}\{i + EB(p_i)\} = 5$, the processing of parent c is considered complete and since there are no more parents, the procedure terminates. The final matching is shown in Figure 26.

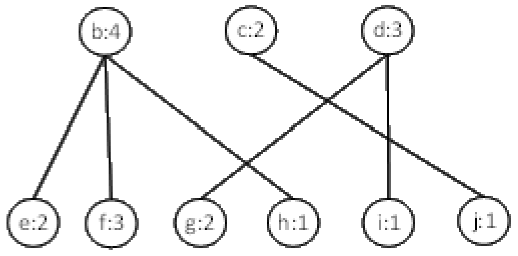


Figure 26 The final matching

3.2 Complexity

The first step of the algorithm is the breadth-first search to construct the layer graph and its time complexity is $O(|V| + |E|) = O(|E|)$.

During the Procedure Matching, first, the parents must be sorted by decreasing number of children. Assuming there are k layers and n_i parents on each layer, where $1 \leq i \leq k$, the complexity of sorting the parents on one layer is $n_i \log n_i$ and the total complexity for all k layers is $\sum_{i=1}^{k} n_i \log n_i$. Since for all $i \in [1, k]$, $n_i \leq |V|$, then of course $\sum_{i=1}^{k} n_i \log n_i \leq \sum_{i=1}^{k} n_i \log|V| = |V| \log|V|$, hence in the worst case this step has $|V| \log|V|$ complexity.

In the second step, the matching requires calculating $EB(p_k)$ for each parent p_k, then finding the parent p_i which gives $max_{1\leq i\leq k}\{i + EB(p_i)\}$ and moving children from one parent to another. From [38] we know that calculating $EB(v)$ has complexity $O\big(deg(v)\big)$, where $deg(v)$ is the degree of vertex v, therefore the total complexity for all parents is $\sum_{v\in V} deg(v) = O(|E|)$. Finding the parent p_i that gives $max_{1\leq i\leq k}\{i + EB(p_i)\}$ has complexity $O(n)$ where n is the number of parents on one layer, hence the total complexity for all layers is the number of vertices in the graph, $O(|V|)$. Moving children from parent p_{max} to parent p_k looks in the worst case at each one of the children of p_k and it searches for a child of p_{max} with the same weight as the respective child of p_k. Each binary search takes at most $O(log|V|)$, and needs to be done for at most all edges, hence the total complexity of this step is $O(|E|\ log|V|)$.

The total complexity of the procedure *Matching* is $O(|V|\,log|V| + |E|\,log|V|) = O(|E|\,log|V|)$.

The last step is the procedure *Broadcast and Improve* which performs a broadcast using the spanning tree generated in the previous step and verifies if idle edges between vertices on the same layer can be used to improve the broadcast time. In the worst case, the procedure goes through all the edges of the graph, hence the complexity of this procedure is $O(|E|)$.

Finally, the total complexity of the algorithm is the sum of the complexities of the procedures above $O(|E|) + O(|E|\,log|V|) = O(|E|\,log|V|)$.

4 Simulation Results and Comparisons with other Heuristics

This chapter focuses on the evaluation of the new 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 [5] (RH)
- The Tree Based Algorithm obtained from [18] (TBA);
- The Random algorithm from [38] (P-R);
- The Semi-Random algorithm [38] (S-R);
- The Minimum-Weight Cover heuristic from [38] (MWC);
- The Minimum-Weight Cover Modified heuristic [38] (MWC-M)

The results are presented in table format with each algorithm on its individual column. In addition to the heuristics 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.[41]

We performed the simulation of the new heuristic 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 new 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).

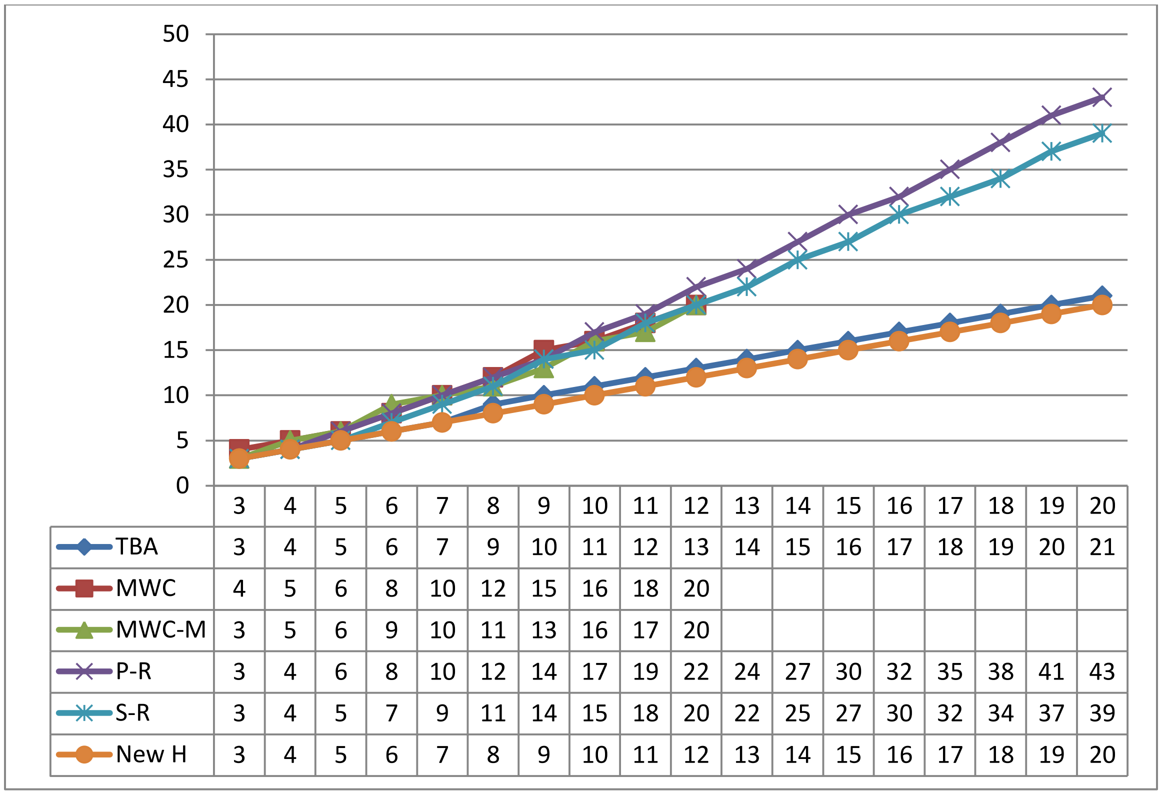
4.1.1 Hypercube

We have already seen above 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 the New Heuristic are presented in Table 1.

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

Table 1 Simulation results in Hypercubes $\mathrm{H_D}$

Figure 27 shows a chart of the simulation results in Hypercubes and we can immediately observe that the New Heuristic provides optimal broadcast time for all dimensions where simulations were run. It clearly performs much better than all the other previous heuristics. The only previous heuristic that has close performance is the Tree Based Algorithm, whose broadcast time is equal to or just 1 more than the optimal broadcast time. With similar time complexity as the Tree Based Algorithm and optimal broadcast time independent of the dimension, we can surely say that the New Heuristic is very suitable for Hypercubes.



	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
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
New H	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

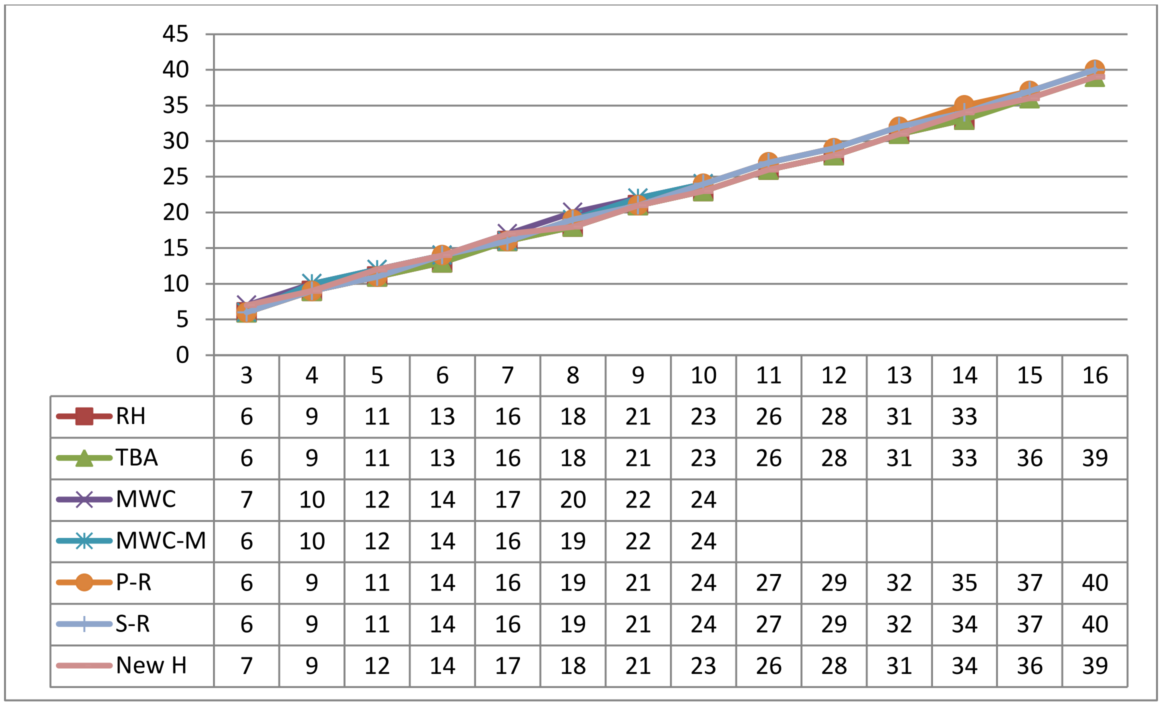
Figure 27 Simulation chart in Hypercubes

4.1.2 Cube Connected Cycles

The theoretical lower and upper bound in Cube Connected Cycles are presented in [24]. The simulation results of the New 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 New Heuristic has similar results with the exception of dimension 5, 6, and 7. 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	New H
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	34
15	36	37	-	36	-	-	37	37	36
16	38	39	-	39	-	-	40	40	39

Table 2 Simulation Results in Cube-Connected Cycles CCC_D



	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RH	6	9	11	13	16	18	21	23	26	28	31	33		
TBA	6	9	11	13	16	18	21	23	26	28	31	33	36	39
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
New H	7	9	12	14	17	18	21	23	26	28	31	34	36	39

Figure 28 Simulation chart in Cube-Connected Cycles

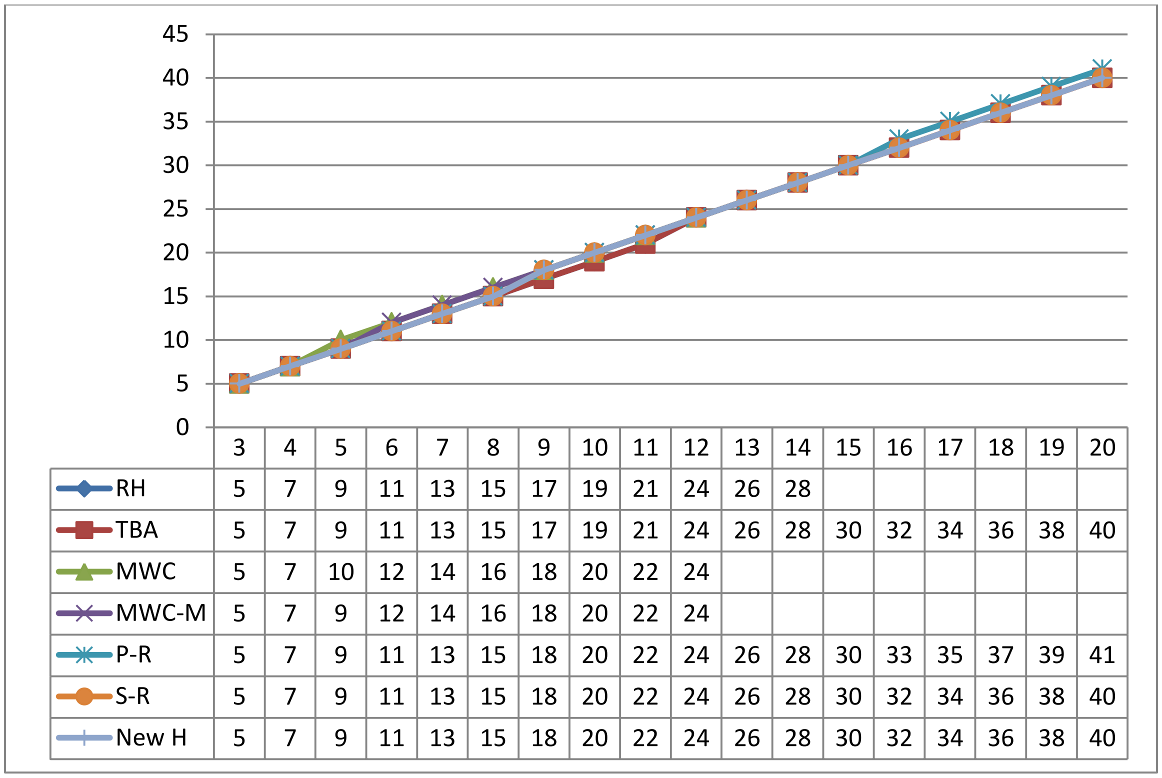
4.1.3 Shuffle-Exchange

The optimal broadcast times in Shuffle-Exchange graphs are presented in [24]. Compared with the previous algorithms with the best performance, the Round Heuristic and the Tree Based Algorithm, the New algorithm has the same performance with the exception of dimensions 9, 10 and 11. When the dimension is less or equal than 8, the resulting broadcast times are optimal. From dimension 9 and up, the broadcast times are always 1 round more than the optimal, for the New Algorithm, as well as for the previous best algorithms, the Round Heuristic and the Tree Based Algorithm. Table 3 shows the simulation results in Shuffle-Exchange graphs.

D	OPT	RH	TBA	MWC	MWC-M	P-R	S-R	New H
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 3 Simulation results of different heuristics in Shuffle-Exchange graphs

In the chart in Figure 29 we notice even better the similar performance of the new algorithm versus previous ones, since the plots of the simulation results are mostly overlapping.



	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
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
S-R	5	7	9	11	13	15	18	20	22	24	26	28	30	32	34	36	38	40
New H	5	7	9	11	13	15	18	20	22	24	26	28	30	32	34	36	38	40

Figure 29 Simulation chart in Shuffle-Exchange graphs

4.1.4 DeBruijn

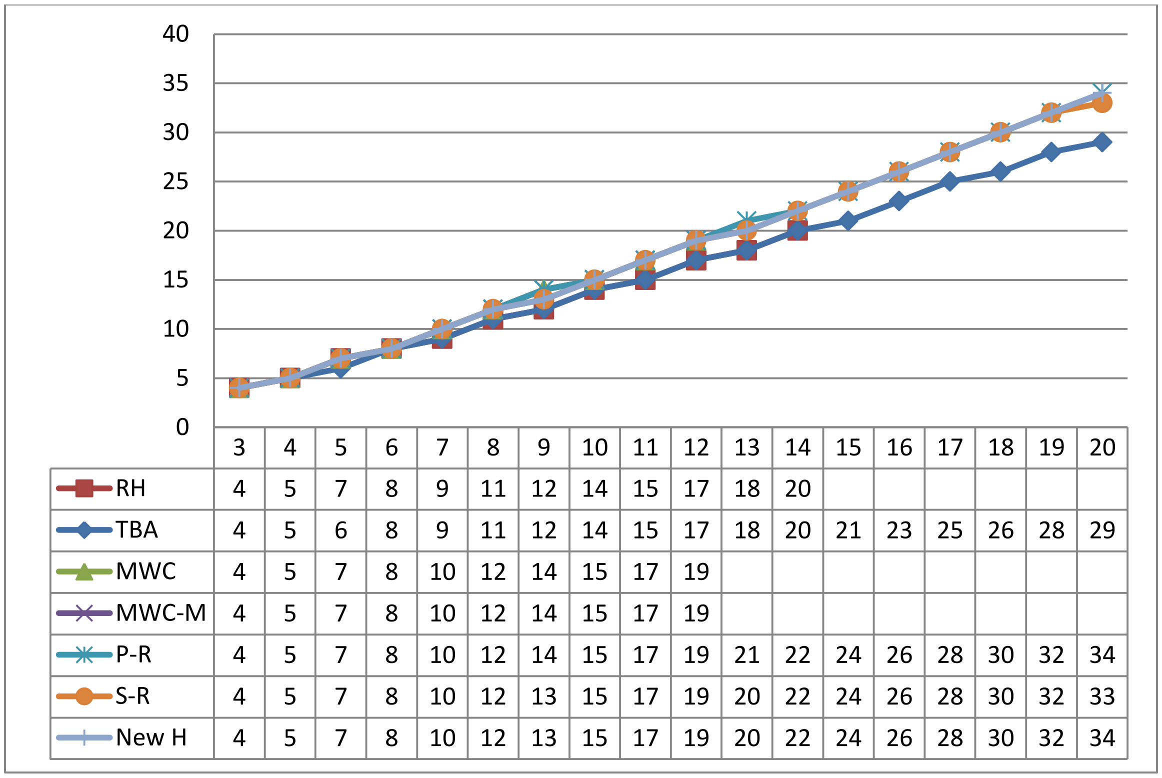
Table 4 shows the simulation results of different heuristics in DeBruijn graphs as well as the lower and upper bounds of DeBruijn graphs. The lower bounds were calculated using the formulas in [27], and they only hold asymptotically. For this reason, Table 4 shows for dimensions 4 and 5 some broadcast times that are less than the given lower bounds.

The simulation results of the New algorithm are the same as the results of the Semi-Random algorithm. For dimensions lower than 17, they do not exceed the upper bound. The Round Heuristic and Tree Based Algorithm have, again, the best results, equal or close to the lower bounds. For most dimensions, the difference between the New algorithm and the best results is at most 2 rounds.

D	LOW	UP	RH	TBA	MWC	MWC-M	P-R	S-R	New H
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	12
9	12	15	12	12	14	14	14	13	13
10	14	17	14	14	15	15	15	15	15
11	15	18	15	15	17	17	17	17	17
12	16	20	17	17	19	19	19	19	19
13	18	21	18	18	-	-	21	20	20
14	19	23	20	20	-	-	22	22	22
15	20	24	-	21	-	-	24	24	24
16	22	26	-	23	-	-	26	26	26
17	23	27	-	25	-	-	28	28	28
18	24	29	-	26	-	-	30	30	30
19	26	30	-	28	-	-	32	32	32
20	27	32	-	29	-	-	34	33	34

Table 4 Simulation results in DeBruijn graphs

The charts in Figure 30 show that the Round Heuristic and the Tree Based Algorithms have the best results and that the results of the New algorithm and the Semi-Random one are overlapping.



	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
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
New H	4	5	7	8	10	12	13	15	17	19	20	22	24	26	28	30	32	34

Figure 30 Simulation chart in DeBruijn graphs

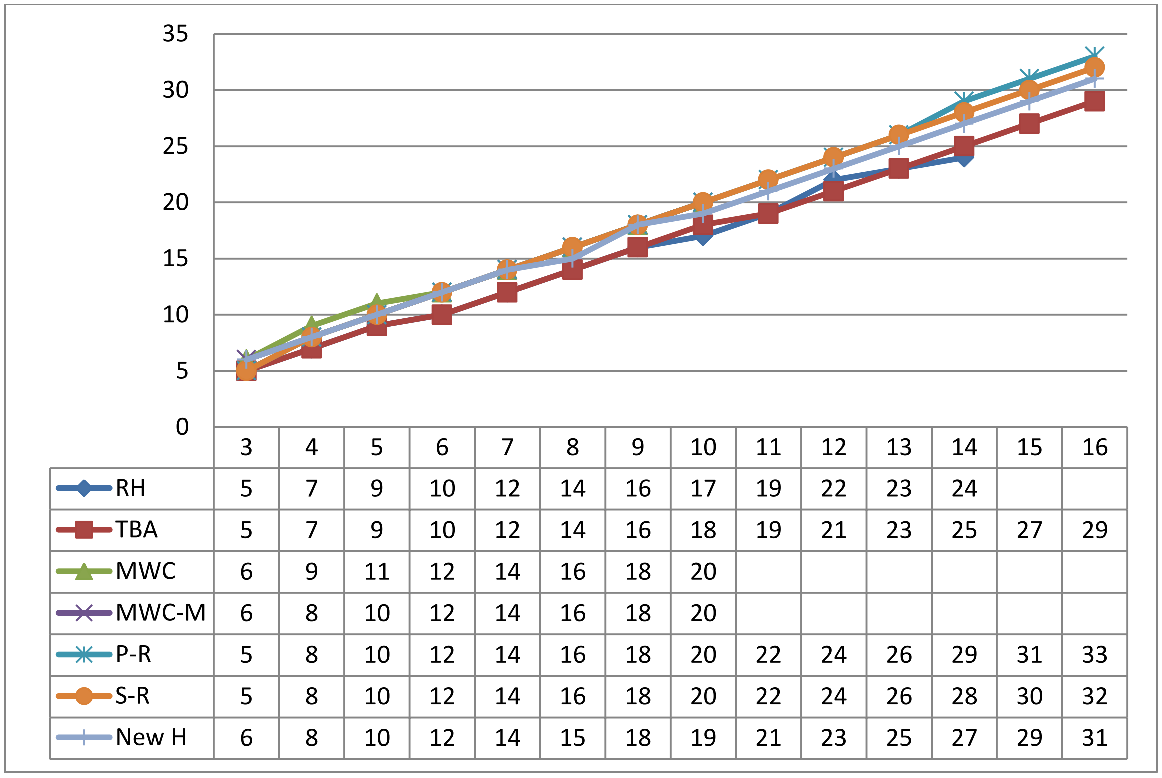
4.1.5 Butterfly

Table 5 shows the simulation results of the different algorithms in Butterfly graphs. The lower and upper bounds are the ones presented in [24]. The New algorithm performs similar to the Semi-Random algorithm for lower dimension, up to 7. As the dimension increases, the New algorithm is consistently 1 round better than the Semi-Random algorithm. The best algorithms are again the Round Heuristic and the Tree Based Algorithm. The difference between the best results and the results of the New algorithm varies between 1 and 3 rounds, increasing with higher dimensions.

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

Table 5 Simulation results in Butterfly graphs

Plotting the simulation results yields the chart in Figure 31, which shows that although the New algorithm performs slightly better than the Semi-Random algorithm, it still does not achieve broadcast times as good as the Round Heuristic or the Tree Based Algorithm.



	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
New H	6	8	10	12	14	15	18	19	21	23	25	27	29	31

Figure 31 Simulation chart in Butterfly graphs

4.2 Other Topologies

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 New algorithm 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 [42], GT-ITM Transit-Stub [42], Tiers [10], and BRITE Top-down Hierarchical models [31]. 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 interconnection 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 Transit-Stub.

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 p. It is obvious that this network model is driven by the probability p. Although this random model does not correspond to any real network, it is still presented and discussed in the network research community.

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 [43]. 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, it 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 32, adapted from [43], shows an example of such a structure.

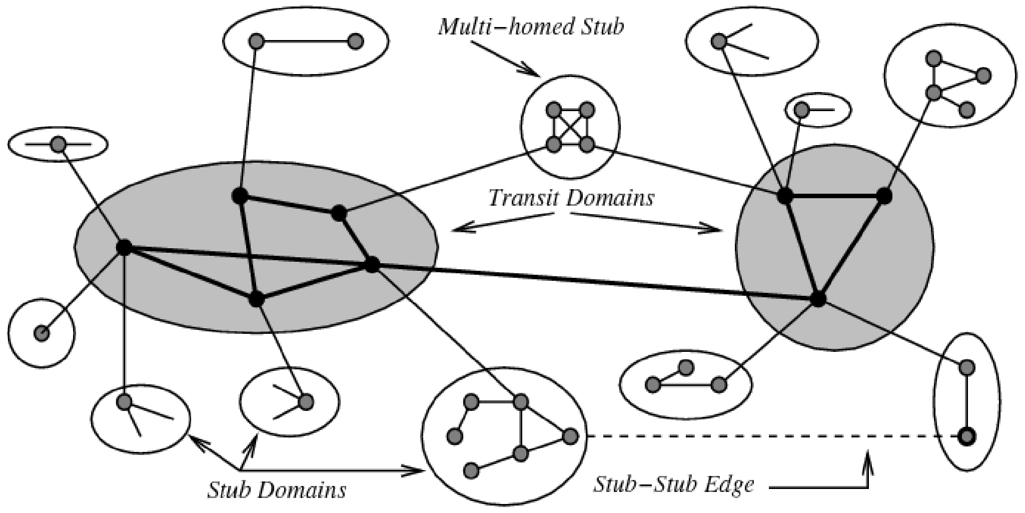


Figure 32 Example of Internet Domain Structure

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
- 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) andN_M, the number of nodes per MAN.
- N_L, the number of LANs per MAN and S_L, the number of nodes per LAN.

The total number of nodes in the graph, N, is given by

$$N = S_W + N_M S_M + N_M N_L S_L$$

The other parameters of the model are:

- The degree of intranetwork 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 33 and Figure 34, adapted from [8], 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.

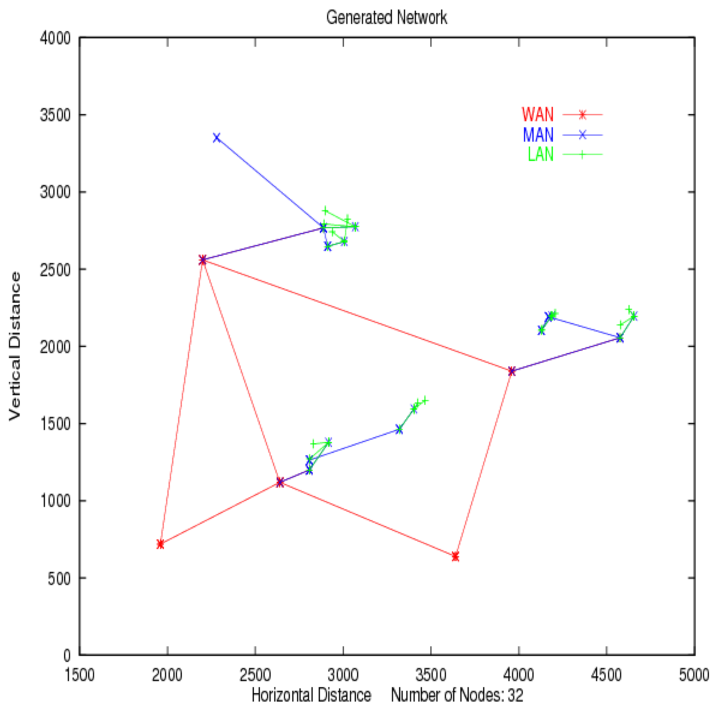


Figure 33 A typical Tiers internetwork

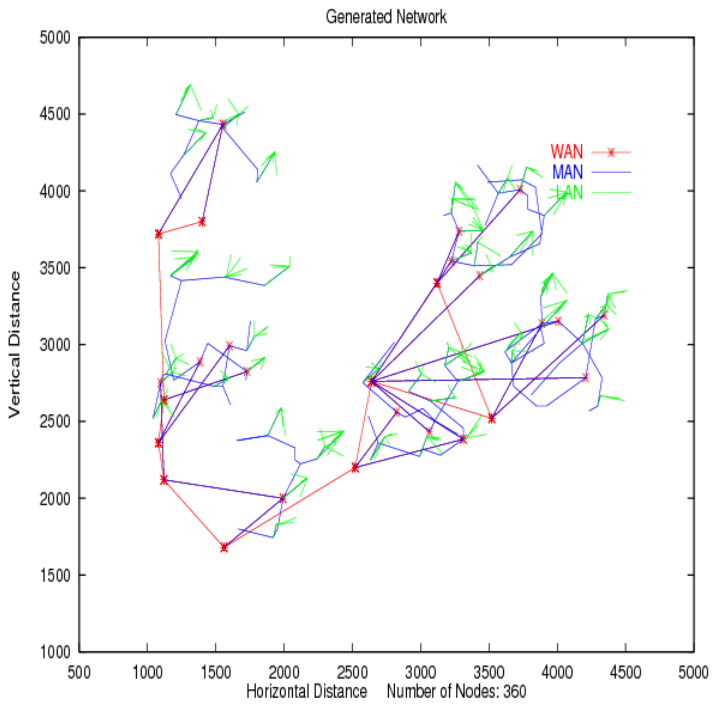


Figure 34 A large Tiers internetwork

GT-ITM Transit-Stub and Tiers implementations generate networks whose topology resembles typical internetworks. Both implementations are based on the explicitly hierarchical modeling approach described in [8]. 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 of Tiers. In both implementations, most of the parameters can be expected to remain constant between runs of generated networks.

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 [31]. 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:

- Using one of the most commonly used models for generating graphs, Waxman's probability model [39], 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 and α and β are parameters in the range $0 < \alpha, \beta \leq 1$.

- Using the Barabasi-Albert (BA) [3] 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 i is added to the network, the probability that it connects to a node j already in the network is given by:

$$\mathrm{P(i,j)} = \frac{\mathrm{d_j}}{\sum_{\mathrm{k} \in \mathrm{V}} \mathrm{d_k}}$$

  where $\mathrm{d_j}$ is the degree of the target node, V is the set of nodes already in the network and $\sum_{\mathrm{k} \in \mathrm{V}} \mathrm{d_k}$ 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.

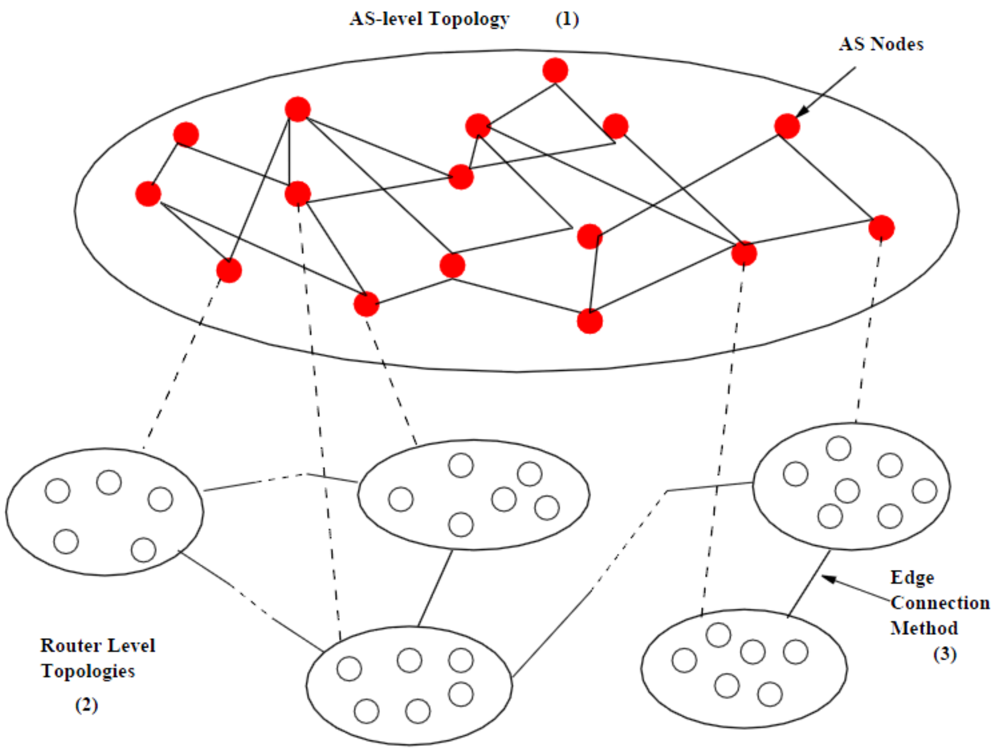


Figure 35 BRITE Top-down hierarchical model

BRITE uses a top-down approach to generate hierarchical topologies. Figure 35 adapted from [31], 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.

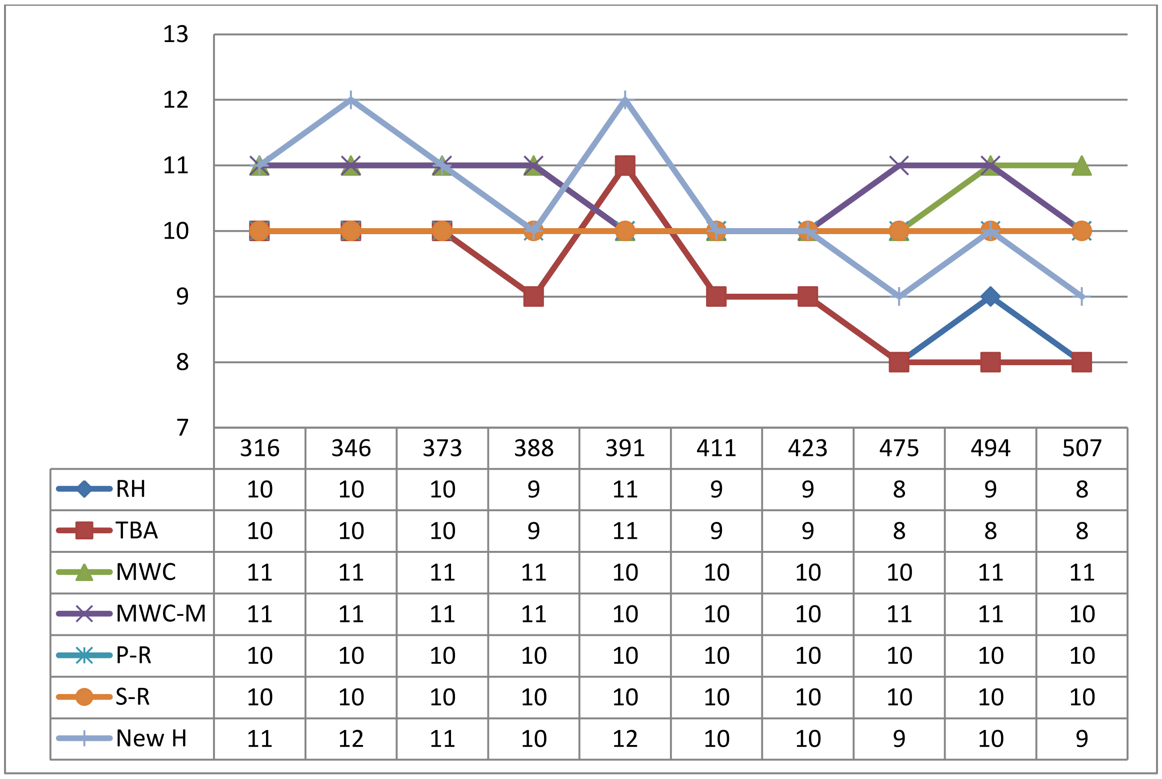
4.2.1 GT-ITM Random Model

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 6 and Figure 36, 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 New heuristic performs slightly worse than previous heuristics, but as the number of edges increases, the New heuristic tends to perform better than all other heuristics except for the Round Heuristic and the Tree Based Algorithm. Compared to these two, the result of the New heuristic are only one round more, but the New Heuristic has the advantage of a much lower time complexity than the Round Heuristic.

P	Edge	RH	TBA	MWC	MWC-M	P-R	S-R	New H
0.015	316	10	10	11	11	10	10	11
0.016	346	10	10	11	11	10	10	12
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	12
0.02	411	9	9	10	10	10	10	10
0.022	423	9	9	10	10	10	10	10
0.024	475	8	8	10	11	10	10	9
0.025	494	9	8	11	11	10	10	10
0.026	507	8	8	11	10	10	10	9

Table 6 Simulation results in GT-ITM Random model with 200 vertices



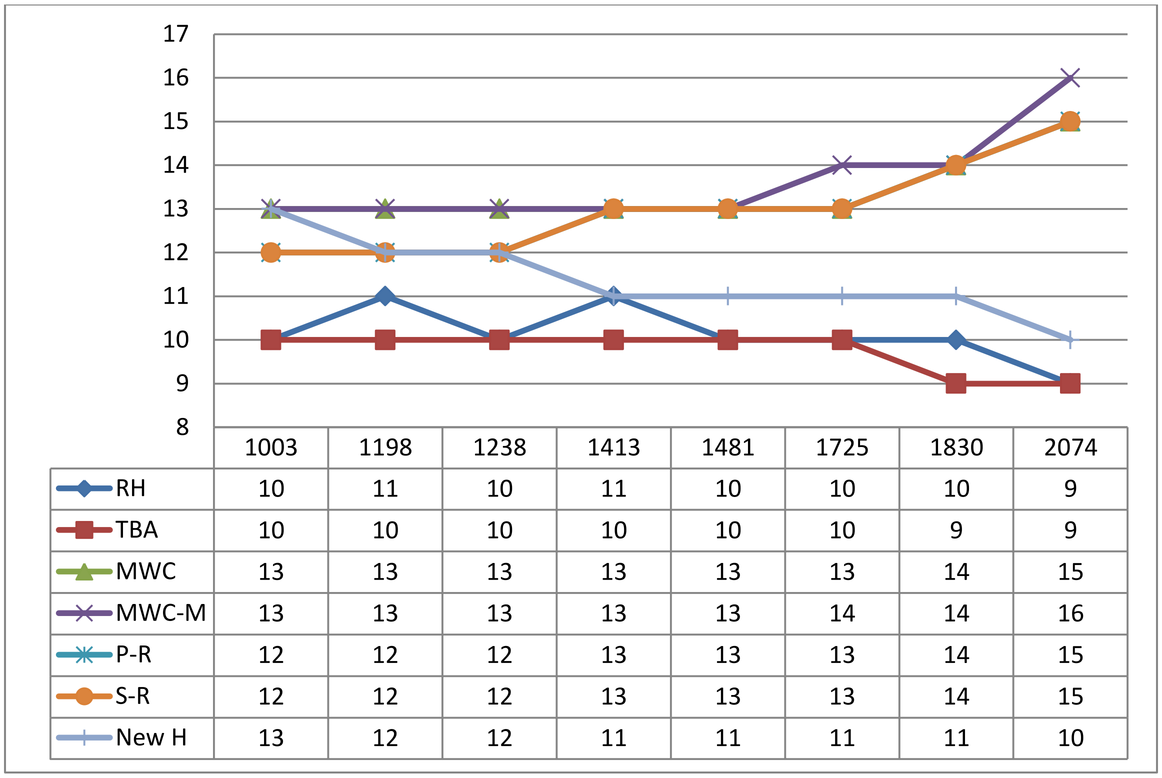
	316	346	373	388	391	411	423	475	494	507
◆ RH	10	10	10	9	11	9	9	8	9	8
■ TBA	10	10	10	9	11	9	9	8	8	8
▲ MWC	11	11	11	11	10	10	10	10	11	11
× MWC-M	11	11	11	11	10	10	10	11	11	10
✳ P-R	10	10	10	10	10	10	10	10	10	10
● S-R	10	10	10	10	10	10	10	10	10	10
+ New H	11	12	11	10	12	10	10	9	10	9

Figure 36 Simulation chart in GT-ITM Random model with 200 vertices

The simulation results in the GT-ITM Random model with 500 vertices are shown in Table 7 and Figure 37. In this case the Tree Based Algorithm has the best results, with the Round Heuristic following closely, whereas the results of all other previous algorithms climb up slowly as the number of edges increases. In contrast, the performance of the New algorithm gets better with the increase in number of edges and when the number of edges is 2074, while the Semi-Random algorithm's results are almost twice those of the Round Heuristic and the Tree Based Algorithm, the performance of the New algorithm gets closer to the best results with only one round more. Compared to the Round Heuristic, the New Heuristic has the advantage of a much lower time complexity; therefore the only previous algorithm with similar time complexity that beats the New heuristic is TBA. However, compared to the TBA algorithm, the New algorithm 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.

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

Table 7 Simulation results in GT-ITM Random model with 500 vertices



	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
+ New H	13	12	12	11	11	11	11	10

Figure 37 Simulation results in GT-ITM Random model with 500 vertices

4.2.2 GT-ITM Transit-Stub Model

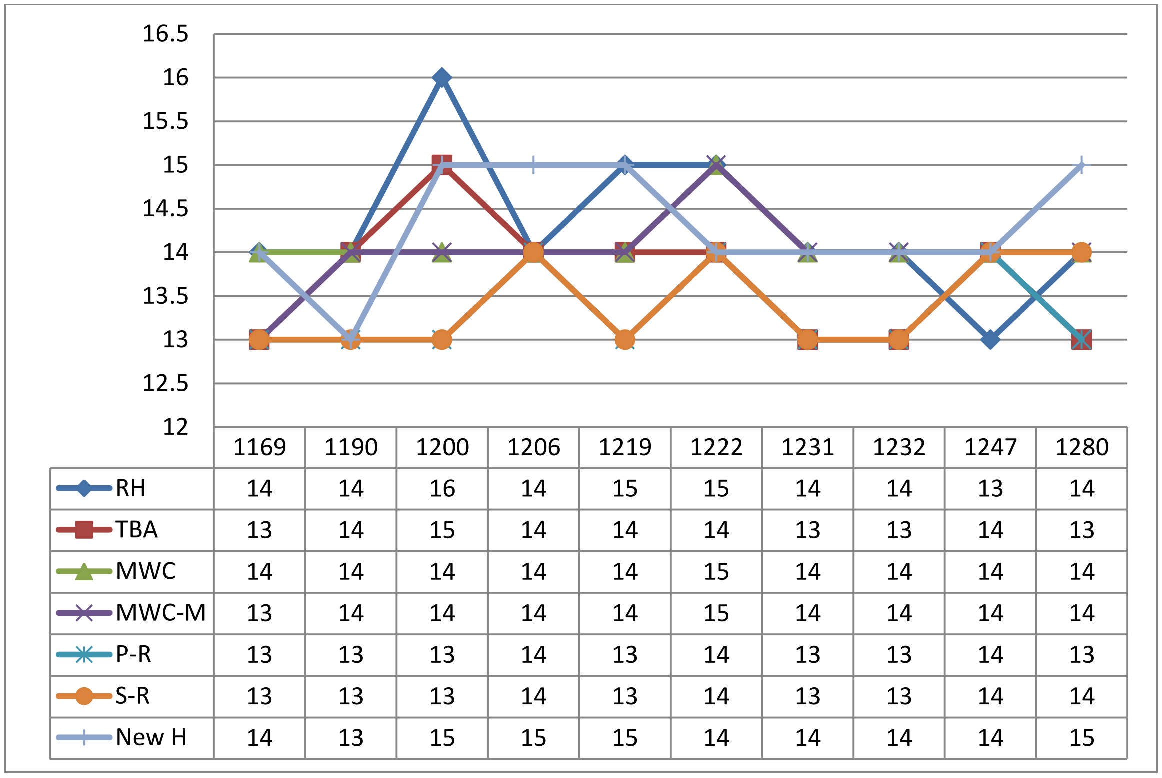
We studied two types of GT-ITM Transit-Stub graphs, one with 600 vertices and the second with 1056 vertices.

The GT-ITM Transit-Stub graphs with 600 vertices were generated using the same parameters used in [38] as follows. The initial seed was 47. 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 8 and Figure 38. 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. The New algorithm 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 New algorithm 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 New algorithm 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.

Edge	RH	TBA	MWC	MWC-M	P-R	S-R	New H
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	15
1222	15	14	15	15	14	14	14
1231	14	13	14	14	13	13	14
1232	14	13	14	14	13	13	14
1247	13	14	14	14	14	14	14
1280	14	13	14	14	13	14	15

Table 8 Simulation results in GT-ITM Transit-Stub model with 600 vertices



	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
+ New H	14	13	15	15	15	14	14	14	14	15

Figure 38 Simulation results in GT-ITM Transit-Stub model with 600 vertices

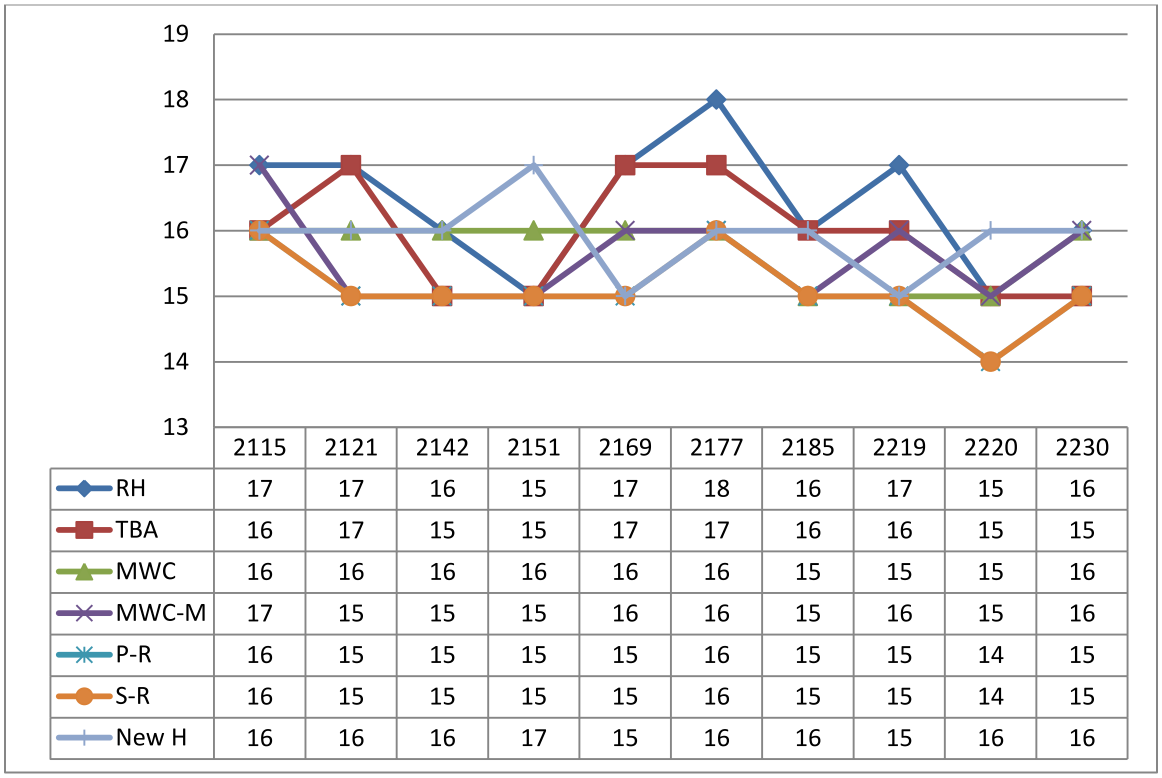
The GT-ITM Transit-Stub graphs with 1056 vertices were generated using the same parameters used in [38] as follows. The initial seed was 47. Each graph had 4 stub domains per transit node, with no extra transit-stub or stub-stub edges. There were 4 transit domains, each of which had 8 nodes, and an edge 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 $4 \times 8 \times (1 + 4 \times 8) = 1056$.

Once again, simulation results in graphs with increasing number of edges are presented below in Table 9 and Figure 39. 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 New algorithm 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 New algorithm 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 New algorithm 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.

Edge	RH	TBA	MWC	MWC-M	P-R	S-R	New H
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	15
2220	15	15	15	15	14	14	16
2230	16	15	16	16	15	15	16

Table 9 Simulation results in GT-ITM Transit-Stub model with 1056 vertices



	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
New H	16	16	16	17	15	16	16	15	16	16

Figure 39 Simulation results in GT-ITM Transit-Stub model with 1056 vertices

4.2.3 Tiers Model

We studied two types of Tiers graphs, one with 355 vertices and the second with 1105 vertices. The parameters for these graphs are the same as in [38] and listed in Table 10 and Table 11. The graphs with 355 vertices have one WAN, ten MANs and five LANs, while graphs of 1105 vertices have one WAN, ten MANs and ten LANs.

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 10 Parameters for Tiers model with 355 vertices

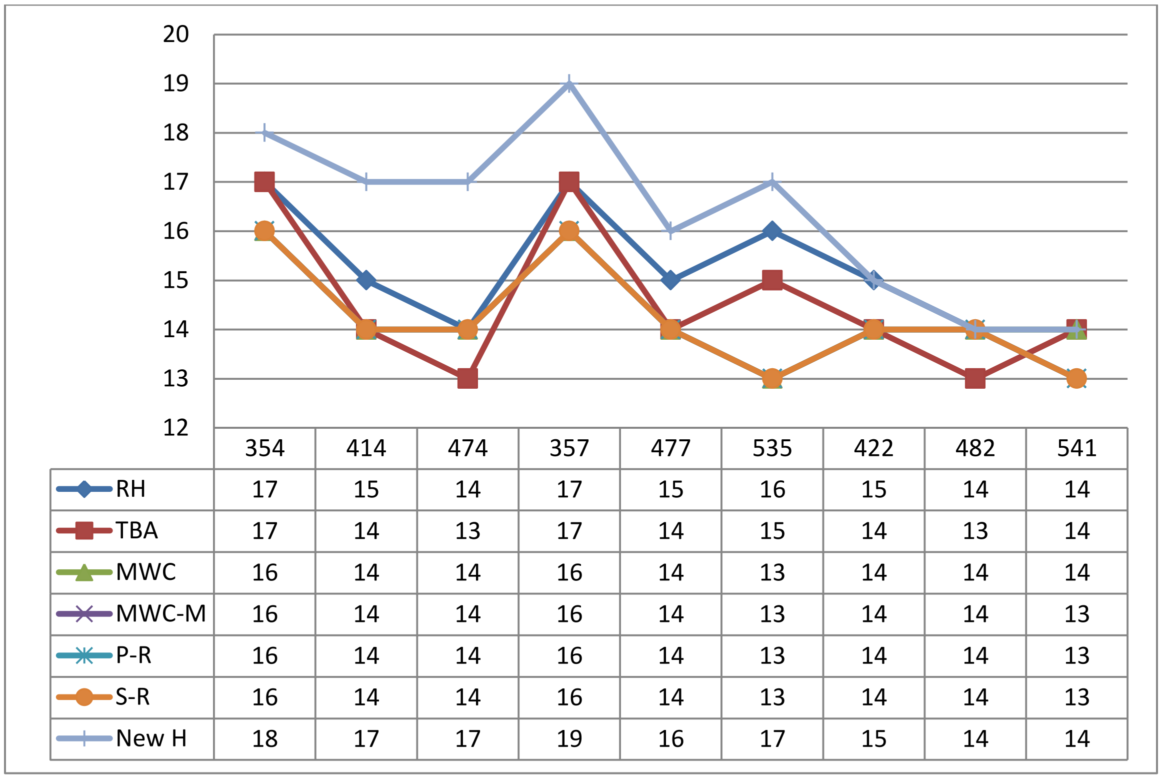
Edge	N_W	N_M	N_L	S_W	S_M	S_L	R_W	R_M	R_L	R_{MW}	R_{LM}
1214	1	10	10	5	10	10	1	1	1	2	2
1324	1	10	10	5	10	10	1	1	1	3	3
1447	1	10	10	5	10	10	1	1	1	4	4
1106	1	10	10	5	10	10	2	2	1	1	1
1216	1	10	10	5	10	10	2	2	1	2	2
1326	1	10	10	5	10	10	2	2	1	3	3
1110	1	10	10	5	10	10	3	2	1	1	1
1220	1	10	10	5	10	10	3	2	1	2	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 11 Parameters for Tiers model with 1105 vertices

In Table 12 and Figure 40 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 New algorithm 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 New algorithm 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.

Edge	RH	TBA	MWC	MWC-M	P-R	S-R	New H
354	17	17	16	16	16	16	18
414	15	14	14	14	14	14	17
474	14	13	14	14	14	14	17
357	17	17	16	16	16	16	19
477	15	14	14	14	14	14	16
535	16	15	13	13	13	13	17
422	15	14	14	14	14	14	15
482	14	13	14	14	14	14	14
541	14	14	14	13	13	13	14

Table 12 Simulation results in Tiers model with 355 vertices



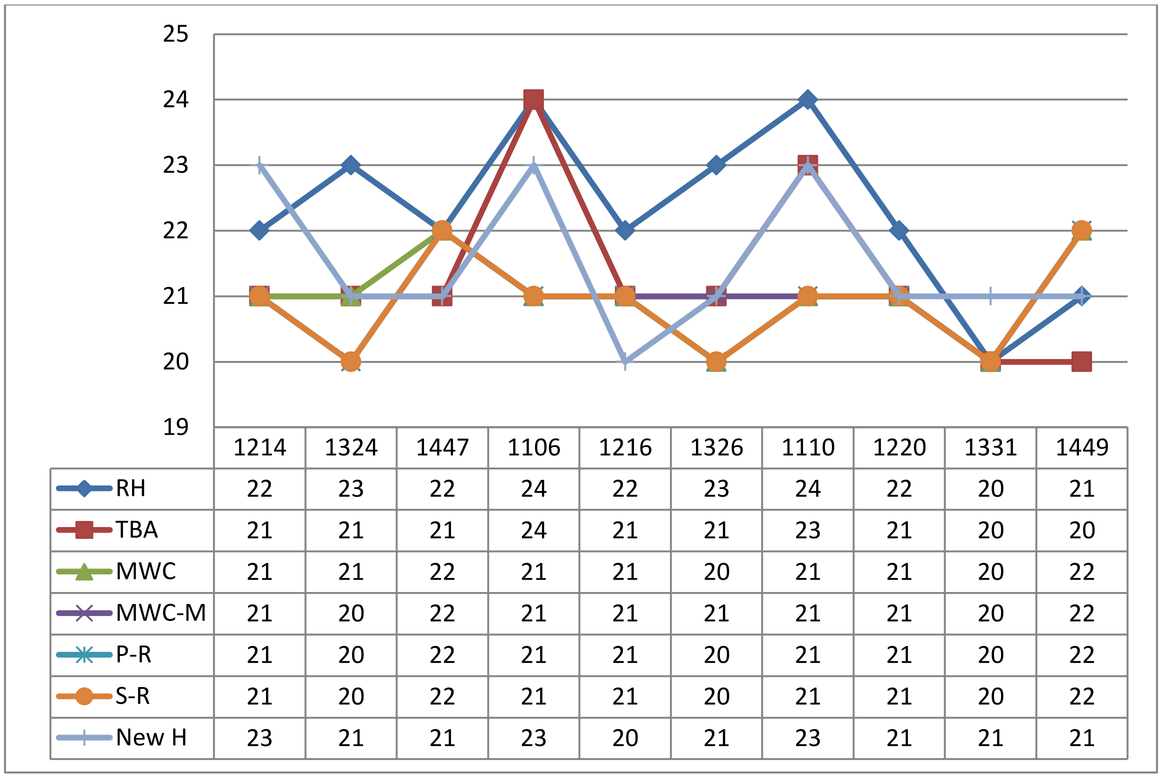
	354	414	474	357	477	535	422	482	541
◆ RH	17	15	14	17	15	16	15	14	14
■ TBA	17	14	13	17	14	15	14	13	14
▲ MWC	16	14	14	16	14	13	14	14	14
× MWC-M	16	14	14	16	14	13	14	14	13
✳ P-R	16	14	14	16	14	13	14	14	13
● S-R	16	14	14	16	14	13	14	14	13
+ New H	18	17	17	19	16	17	15	14	14

Figure 40 Simulation results in Tiers model with 355 vertices

In Table 13 and Figure 41 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 New algorithm gives the best results in three graphs, the ones with 1447, 1216, and 1220 edges. However, compared to the Semi-Random algorithm, the New algorithm 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 New algorithm 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.

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

Table 13 Simulation results in Tiers model with 1105 vertices



	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
New H	23	21	21	23	20	21	23	21	21	21

Figure 41 Simulation results in Tiers model with 1105 vertices

4.2.4 BRITE Top-down Hierarchical Model

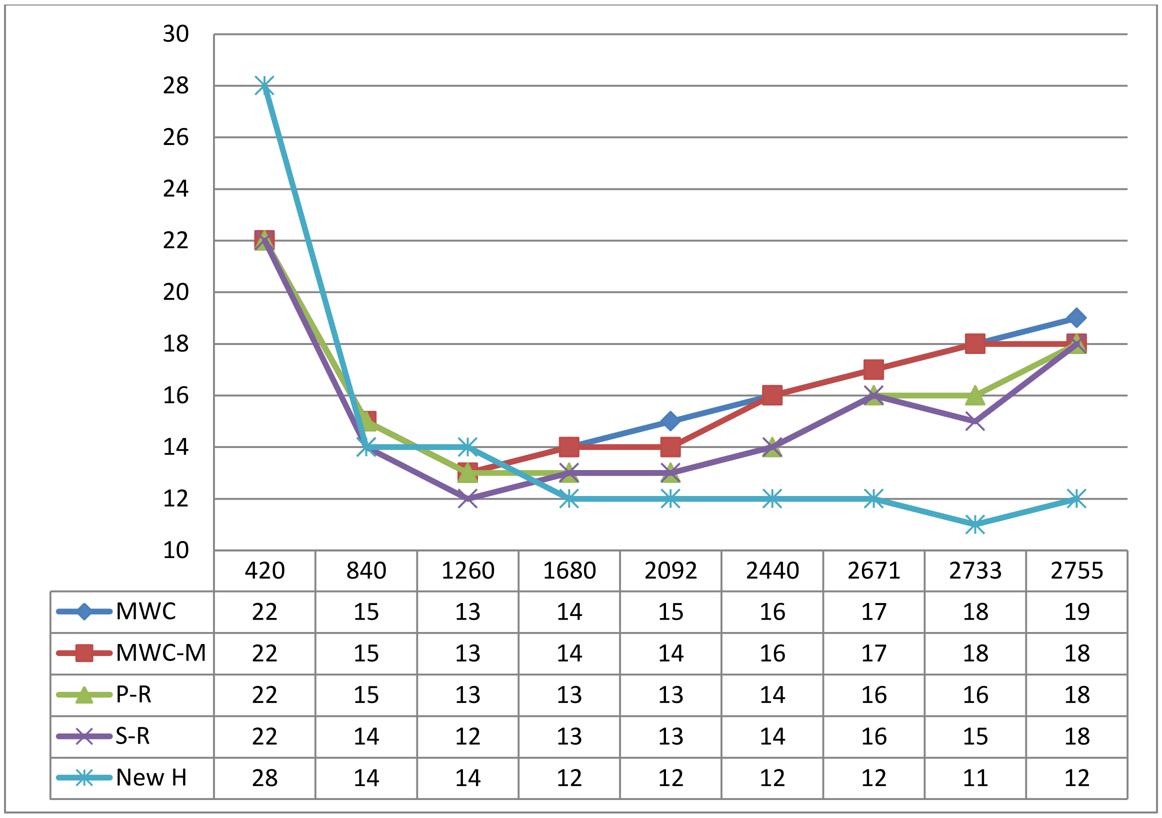
In this section we present our simulation results in Top-down hierarchical model graphs generated with the BRITE topology generator. Four types of graphs were studied, two with 400 vertices and two with 1000 vertices, each one constructed with Waxman and Barabasi-Albert models. Since there are no previous results in these topologies for the Round Heuristic and Tree Based Algorithm, the results of the New algorithm are compared only with the other four (P-R, S-R, MWC and MWC-Modified). The configurations of the graphs studied uses the same parameters as in [38].

In the BRITE Top-down models with 400 vertices the number of vertices at 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. The parameters of the Waxman model are $\alpha = 0.15$ and $\beta = 0.2$.

In Table 14 and Figure 42 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 new algorithm 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.

Edge	MWC	MWC-M	P-R	S-R	New H
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	12
2733	18	18	16	15	11
2755	19	18	18	18	12

Table 14 Simulation results in BRITE Top-down Waxman model with 400 vertices



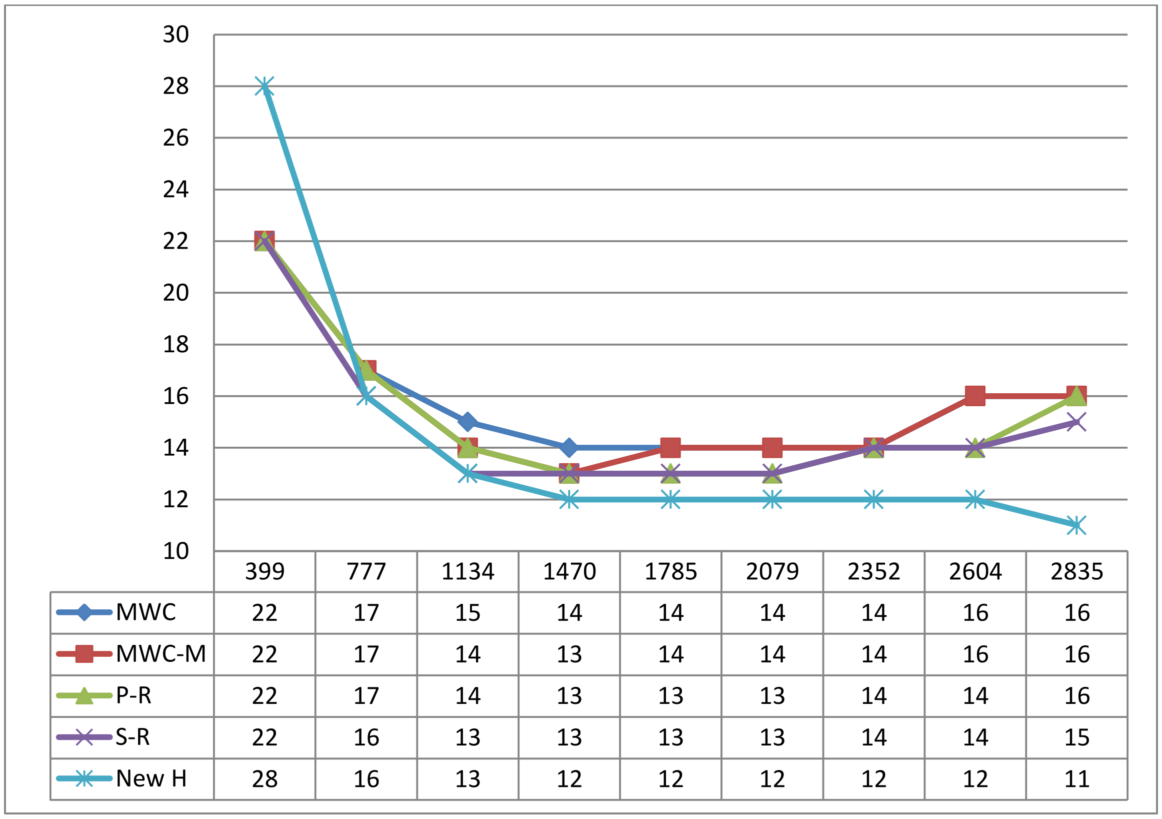
	420	840	1260	1680	2092	2440	2671	2733	2755
◆ MWC	22	15	13	14	15	16	17	18	19
■ MWC-M	22	15	13	14	14	16	17	18	18
▲ P-R	22	15	13	13	13	14	16	16	18
× S-R	22	14	12	13	13	14	16	15	18
✳ New H	28	14	14	12	12	12	12	11	12

Figure 42 Simulation results in BRITE Top-down Waxman model with 400 vertices

In Table 15 and Figure 43 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 New algorithm 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 New Algorithm is better than the next one in performance, is quite significant.

Edge	MWC	MWC-M	P-R	S-R	New H
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 15 Simulation results in BRITE Top-down BA model with 400 vertices



	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
New H	28	16	13	12	12	12	12	12	11

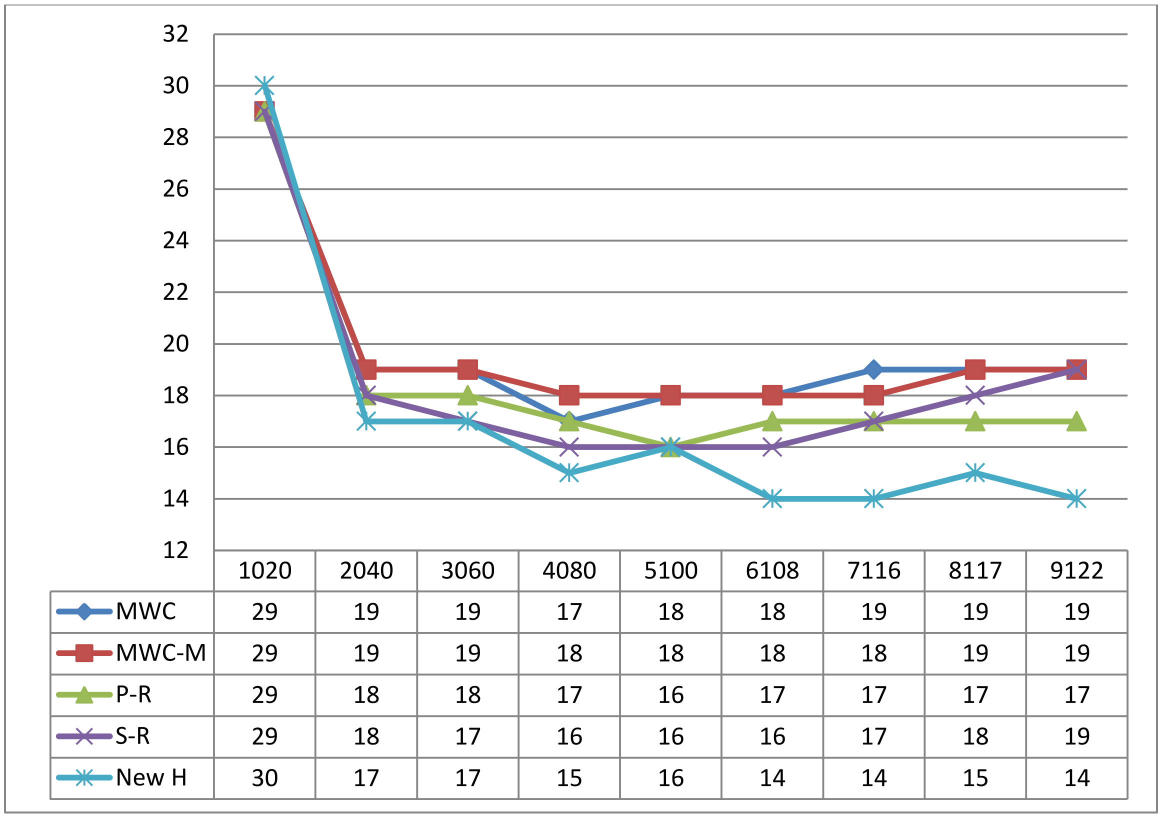
Figure 43 Simulation results in BRITE Top-down BA model with 400 vertices

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 16 and Figure 44 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 New algorithm improves and for the graphs with high number of edges it gets 2 or 3 rounds better than the next best results.

Edge	MWC	MWC-M	P-R	S-R	New H
1020	29	29	29	29	30
2040	19	19	18	18	17
3060	19	19	18	17	17
4080	17	18	17	16	15
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 16 Simulation results in BRITE Top-down Waxman model with 1000 vertices



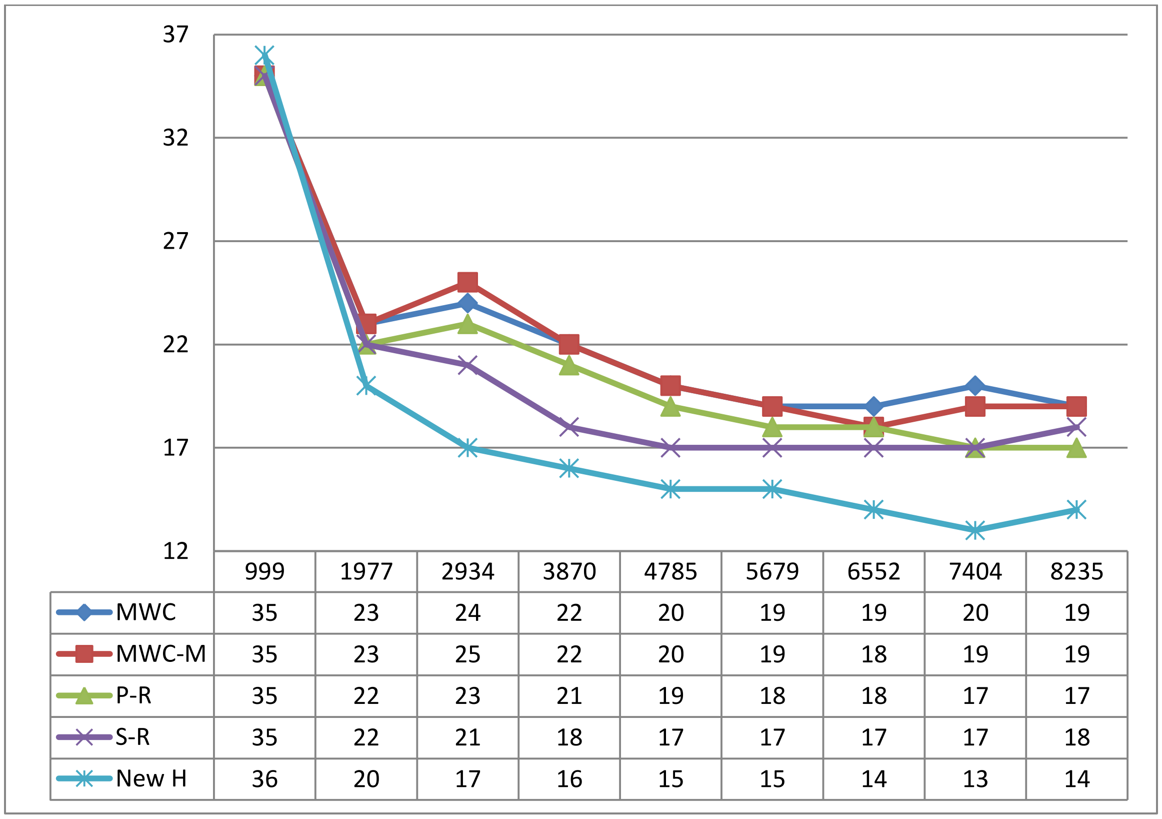
	1020	2040	3060	4080	5100	6108	7116	8117	9122
◆ MWC	29	19	19	17	18	18	19	19	19
■ MWC-M	29	19	19	18	18	18	18	19	19
▲ P-R	29	18	18	17	16	17	17	17	17
× S-R	29	18	17	16	16	16	17	18	19
✳ New H	30	17	17	15	16	14	14	15	14

Figure 44 Simulation results in BRITE Top-down Waxman model with 1000 vertices

In Table 17 and Figure 45 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 New algorithm beats the next best results of the previous algorithms by at least 2 rounds, and in a couple of graphs by 4 rounds.

Edge	MWC	MWC-M	P-R	S-R	New H
999	35	35	35	35	36
1977	23	23	22	22	20
2934	24	25	23	21	17
3870	22	22	21	18	16
4785	20	20	19	17	15
5679	19	19	18	17	15
6552	19	18	18	17	14
7404	20	19	17	17	13
8235	19	19	17	18	14

Table 17 Simulation results in BRITE Top-down BA model with 1000 vertices



	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
New H	36	20	17	16	15	15	14	13	14

Figure 45 Simulation results in BRITE Top-down BA model with 1000 vertices

5 Conclusion and Future Work

We set out to examine and improve upon algorithms for broadcasting in arbitrary graphs. We have accomplished the following.

Given that determining the broadcast time for an arbitrary vertex u in an arbitrary graph G is NP-complete, we surveyed existing approximation and heuristic algorithms and analyzed their behavior in both commonly used topologies and other topologies used to study networking algorithms.

Since generating example networks is important for testing broadcasting algorithms, we described some of the difficulties of modeling the topology of typical communications networks and provided brief details for different modeling approaches and several implementations popular in the research community.

We also proposed a New heuristic for broadcasting in arbitrary networks. Based on the layer graph, the New heuristic first generates a spanning tree using a new matching strategy between each pair of adjacent layers. The new matching strategy is a greedy strategy, which tries to locally optimize the broadcast time between each pair of adjacent layers hoping this would yield a close to optimum global broadcast time. In the final step, the broadcast scheme is improved by considering vertices that are idle at any round and potential edges that were removed from the initial graph during the construction of the layer graph. The final broadcast scheme 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.

The New heuristic outperforms previous heuristics in Hypercubes, where it produces optimal broadcast time. It also produces optimal broadcast time in most of the dimensions simulated in Cube Connected Cycles, matching the performance of the best previous results. Previous performance results in Shuffle-Exchange graphs are also matched.

Looking at network topologies that mimic the Internet model, the performance of the New algorithm varies from model to model. In BRITE Top-down hierarchical model topologies the results are much better than previous heuristics. The new algorithm, 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. In Tiers models, it performs poorly for graphs with low number of edges, but starts catching up to the best results as the number of edges increases.

The other advantage of the New algorithm is its low time complexity of $O(|E|\ log|V|)$, which is very close to $O(|E|)$, the lowest complexity of some of the other heuristics mentioned in this thesis. Still, the new heuristic has comparable or even better broadcast times.

From the current results, one can see that a local matching strategy, layer by layer in the layer graph does not yield the best results in all topologies, so future improvements could be made so that the layer by layer matching strategy is made more global, considering also what happens in upper layers of the layer graph. Looking at all the heuristics discussed in this thesis, we can also observe that it would probably be difficult to design a new heuristic with better performance because some of the existing heuristics already achieve the optimal broadcast time $log|V|$ in some networks. Future work might also approach the problem from a different angle and try to design an approximation algorithm for the problem of broadcast time in arbitrary graphs.

References

1. W. Aiello, F. Chung, and L. Lu. *Random evolution in massive graphs*, in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, 2001. IEEE. p. 510-519.

2. A. Bar-Noy, S. Guha, J. Naor, and B. Schieber, *Message Multicasting in Heterogeneous Networks.* SIAM Journal on Computing, 2000. **30**(2): p. 347-358.

3. A.-L. Barabási and R. Albert, *Emergence of Scaling in Random Networks.* Science, 1999. **286**(5439): p. 509-512.

4. D. Barth and P. Fraigniaud, *Approximation algorithms for structured communication problems*, in *Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, 1997, ACM: Newport, Rhode Island, USA. p. 180-188.

5. R. Beier and J.F. Sibeyn, *A powerful heuristic for telephone gossiping,* in *Proceedings of Seventh International Colloquium on Structural Information and Communication Complexity, SIROCCO 2000,* p. 17-36.

6. J.-C. Bermond and C. Peyrat. *Broadcasting in de Bruijn networks*, in *Proc. 19th SE Conference on Combinatorics, Congressus Numerantium*, 1988. p. 283-292.

7. J.-C. Bermond, P. Hell, A.L. Liestman, and J.G. Peters, *Broadcasting in bounded degree graphs.* SIAM Journal on Discrete Mathematics, 1992. **5**(1): p. 10-24.

8. K.L. Calvert, M.B. Doar, and E.W. Zegura, *Modeling Internet topology.* Communications Magazine, IEEE, 1997. **35**(6): p. 160-163.

9. M.J. Dinneen, *The complexity of broadcasting in bounded-degree networks.* Computer Research and Applications, 1994.

10. M.B. Doar. *A better model for generating test networks*, in *Global Telecommunications Conference*, 1996. IEEE. p. 86-93.

11. M. Elkin and G. Kortsarz, *Sublogarithmic approximation for telephone multicast: path out of jungle (extended abstract)*, in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, 2003, Society for Industrial and Applied Mathematics: Baltimore, Maryland. p. 76-85.

12. M. Elkin and G. Kortsarz, *A Combinatorial Logarithmic Approximation Algorithm for the Directed Telephone Broadcast Problem.* SIAM Journal on Computing, 2005. **35**(3): p. 672-689.

13. A. Farley and S. Hedetniemi. *Broadcasting in grid graphs*, in *Proc. 9th SE Conf. Combinatorics, Graph Theory, and Computing, Utilitas Mathematica*, 1978. p. 275-288.

14. A. Farley, S. Hedetniemi, S. Mitchell, and A. Proskurowski, *Minimum broadcast graphs.* Discrete Mathematics, 1979. **25**(2): p. 189-193.

15. P. Fraigniaud and E. Lazard, *Methods and problems of communication in usual networks.* Discrete Applied Mathematics, 1994. **53**(1–3): p. 79-133.

16. P. Fraigniaud and J.G. Peters, *Minimum linear gossip graphs and maximal linear (Δ, k)-gossip graphs.* Networks, 2001. **38**(3): p. 150-162.

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

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

19. H.A. Harutyunyan and W. Wei. *Broadcasting Algorithm Via Shortest Paths*, in *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, 2010. p. 299-305.

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

21. C.J. Hoelting, D.A. Schoenefeld, and R.L. Wainwright, *A genetic algorithm for the minimum broadcast time problem using a global precedence vector*, in *Proceedings of the 1996 ACM symposium on Applied Computing*, 1996, ACM: Philadelphia, Pennsylvania, USA. p. 258-262.

22. J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, *Dissemination of Information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-Tolerance (Texts in Theoretical Computer Science. An EATCS Series).* 2005.

23. J. Hromkovič, C.-D. Jeschke, and B. Monien, *Optimal algorithms for dissemination of information in some interconnection networks.* Algorithmica, 1993. **10**(1): p. 24-40.

24. J. Hromkovič, R. Klasing, B. Monien, and R. Peine, *Dissemination of information in interconnection networks (broadcasting & gossiping)*, in *Combinatorial network theory*. 1996, Springer. p. 125-212.

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

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

27. R. Klasing, B. Monien, R. Peine, and E.A. Stöhr, *Broadcasting in butterfly and deBruijn networks.* Discrete Applied Mathematics, 1994. **53**(1–3): p. 183-197.

28. G. Kortsarz and D. Peleg, *Approximation Algorithms for Minimum-Time Broadcast.* SIAM Journal on Discrete Mathematics, 1995. **8**(3): p. 401-427.

29. F. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. 1992*, Morgan Kaufmann, San Mateo, CA.

30. A. Liestman and J. Peters, *Broadcast Networks of Bounded Degree.* SIAM Journal on Discrete Mathematics, 1988. **1**(4): p. 531-540.

31. A. Medina, A. Lakhina, I. Matta, and J. Byers. *BRITE: an approach to universal topology generation*, in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, 2001. p. 346-353.

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

33. R. Ravi. *Rapid rumor ramification: approximating the minimum broadcast time*, in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, 1994. p. 202-213.

34. P. Scheuermann and M. Edelberg, *Optimal broadcasting in point-to-point computer networks.* Dep. Elec. Eng. Comput. Sci., Northwestern Univ., Evanston, IL, Tech. Rep, 1981.

35. P. Scheuermann and G. Wu, *Heuristic algorithms for broadcasting in point-to-point computer networks.* Computers, IEEE Transactions on, 1984. **100**(9): p. 804-811.

36. C. Schindelhauer, *On the inapproximability of broadcasting time.* Approximation Algorithms for Combinatorial Optimization, 2000: p. 226-237.

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

38. W. Wang, *Heuristics for Message Broadcasting in Arbitrary Networks*, Masters Thesis, in Computer Science, Concordia University. 2010, p. 86.

39. B.M. Waxman, *Routing of multipoint connections.* Selected Areas in Communications, IEEE Journal on, 1988. **6**(9): p. 1617-1622.

40. Wikipedia. *Network topology*. Available from: http://en.wikipedia.org/wiki/Network_topology, last visited 2013 March, 2nd.

41. Wikipedia. *Confidence Interval*. Available from: http://en.wikipedia.org/wiki/Confidence_interval, last visited 2013 July, 2nd.

42. 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*, 1996. IEEE. p. 594-602.

43. E.W. Zegura, K.L. Calvert, and M.J. Donahoo, *A quantitative comparison of graph-based models for Internet topology.* IEEE/ACM Transactions on Networking (TON), 1997. **5**(6): p. 770-783.