HEURISTICS FOR MESSAGE BROADCASTING IN ARBITRARY NETWORKS

WEI WANG

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

MONTRÉAL, QUÉBEC , CANADA

JUNE 2010

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Wei Wang

Entitled: Heuristics 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. David Ford

______________________________________ Examiner
Dr. Thomas Fevens

______________________________________ Examiner
Dr. Dhrubajyoti Goswami

______________________________________ Supervisor
Dr. Hovhannes Harutyunyan

Approved by ____
Chair of Department or Graduate Program Director

________20____ ____
Dr. Robin Drew, Dean
Faculty of Engineering and Computer Science

Abstract

Heuristics for Message Broadcasting in Arbitrary Network

Wei Wang

With the increasing popularity of interconnection networks, efficient information dissemination has become a popular research area. Broadcasting is one of the information dissemination primitives. Finding the optimal broadcasting scheme for any originator in an arbitrary network has been proved to be an NP-Hard problem. In this thesis, two new heuristics that generate broadcast schemes in arbitrary networks are presented. Both of them have $O(|E|)$ time complexity. Moreover, in the broadcast schemes generated by the heuristics, each vertex in the network receives the message via a shortest path. Based on computer simulations of these heuristics in some commonly used topologies and network models, and comparing the results with the best existing heuristics, we conclude that the new heuristics show comparable performances while having lower complexity.

Acknowledgements

I would like to sincerely thank my supervisor, Dr. H. Harutyunyan, for his sagacious instructions and continuous support through my study life at Concordia. I would also thank Mr. Edward Marashlian for his valuable comments on this thesis.

Contents

List of Figures

List of Tables

1 Introduction

Since the birth of computers, innumerable endeavors have been dedicated to speed up the process of computing. With the development of materials and very-large-scale integration, multi computer and multi processor systems have been turned into a mature technology. Along with the improvement on the physical level, it is essential to design more efficient algorithms for information distribution to get the most benefit out of the advances in the hardware domain. Nowadays, this is an especially crucial problem for parallel computing and distributed systems. Moreover, coming into 21st century, different kinds of networks, such as internet and virtual social networks, have become a main part of people's daily life. One way to perform efficient information dissemination is to compress the amount of data being transferred. The second approach is to minimize the delay of information spreading [37], including designing topologies with optimal dissemination time and efficient algorithms for message distribution.

Broadcasting is a kind of communication process, which spreads the message from the originator to the rest of the network. Nowadays, many fields have seen the contributions of the research of broadcasting, such as parallel computing, management of distributed systems, communication among multi processors, internet messaging and anti virus spreading, etc. In this thesis we focus on the algorithms and heuristics for broadcasting in arbitrary networks, and two new heuristics for broadcasting in arbitrary networks are introduced. These heuristics have low complexity and always spread the messages through shortest paths.

1.1 Models of Broadcasting

There are four communication primitives:

- Routing, one to one communication;
- Broadcasting, one to all communication;
- Multicasting, one to multiple communication;
- Gossiping, all to all communication.

In this thesis, we will be concerned only with broadcasting, which is the process of information dissemination in an interconnection network by which messages are transmitted from the originator to the rest of the network.

Broadcasting is assumed to take place in discrete time units. Each time unit is also called a round. During each round, a series of calls, i.e., message-exchanging between adjacent vertices in the network, take place in parallel. The total number of rounds after which all nodes of the network are informed is used to measure the broadcast time.

In order to simplify the analysis of a broadcast process, we assume that broadcasting is done according to the following constraints. This broadcast model is known as the classical or original model.

- Each call involves only one informed vertex and one of its uninformed neighbors.
- Each call costs one unit of time, i.e., one round.

- A vertex can participate in only one call per unit of time.
- In one unit of time, many calls can be performed in parallel.

At the beginning of broadcasting, only one vertex u, called the *originator*, is informed. During each round, all informed vertices transfer the message to one of their uninformed neighbors. A *call* is the action of transfering the message from an informed vertex to an uninformed neighbor, and the broadcasting scheme is composed of a sequence of parallel calls.

Figure 1 shows a simple broadcast scheme in three rounds, which is presented in the labeling of the edges. Vertex o, with the shadowed background, is the originator, which means it has one message to broadcast. During each round, the message is passed to some uninformed vertices. Finally, after three rounds, all six vertices in the network possess the message.

In general, any interconnection network can be regarded as a connected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges in the graph G. We say vertex $u \in V$ and vertex $v \in V$ are neighbors, if there is an edge $e \in E$ between u and v such

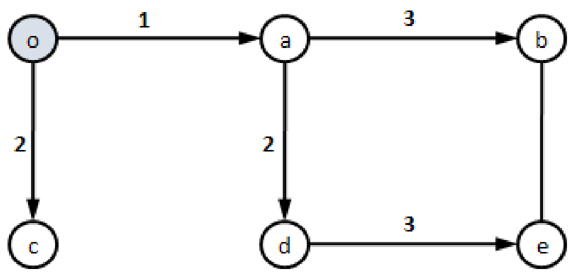


Figure 1: Broadcast scheme in a network with 6 vertices.

that $e = (u, v)$. The *degree* of a vertex v, denoted by $deg(v)$, is the number of neighbors of v. The degree of a graph G, denoted by Δ, is the maximum degree among all vertices in graph G, i.e., $\Delta = \max\{deg(v), v \in V\}$. A *path* p in a graph is a sequence of vertices and edges in the form $p = v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n$, where $e_i = (v_i, v_{i+1})$. Along the path p, a message can travel from vertex v_1 to vertex v_n. The number of edges in a path is called the *length* of the path. The length of the shortest path between two vertices u and v is their *distance*, denoted by $dist(u, v)$. The *diameter* of a graph G, denoted by $D(G)$,is the longest distance between any pair of vertices in the graph. A graph G is called a *connected* graph, if there exists at least one path between any pair of vertices of graph G.

Given an originator u, we define the *broadcast time*, $b(u)$, as the minimum number of time units required to complete broadcasting from vertex u. It is easy to conclude that for any vertex u in a connected network G with n vertices, $\lceil \log n \rceil \leq b(u) \leq n - 1$, since during each time unit the number of informed vertices can at most double. The broadcast time $b(G)$ of the network G is defined as the *maximum broadcast time* among all the vertices, $b(G) = \max\{b(u) | u \in V\}$. Some results of related problems have been presented in papers[21, 14, 24, 25].

1.2 NP-Completeness

The abbreviation NP refers to “nondeterministic polynomial time”. NP is the set of decision problems solvable in polynomial time by a non-deterministic Turing machine.[42]

NP-complete is a class of problems having two properties:

- Any given solution to the problem can be verified quickly (in polynomial time); the set of problems with this property is called NP (nondeterministic polynomial time).
- If the problem can be solved quickly (in polynomial time), then so can every problem in NP [43], i.e., every problem in NP can be reduced to it in polynomial time.

The broadcast problem, determining $b(u)$ for an arbitrary originator u in an arbitrary graph G, is NP-complete, that is proved in [40]. From [18], we know that the three-dimensional matching (3DM) problem is NP-complete, and in [40], the 3DM problem is shown to be reducible to the broadcast problem in polynomial time. In the rest of this section we present a sketch of the proof of the NP completeness of the broadcast problem.

The *3DM* problem is defined as follows. Given sets $X = \{x_1, x_2, \ldots, x_m\}$, $Y = \{y_1, y_2, \ldots, y_m\}$, $Z = \{z_1, z_2, \ldots, z_m\}$, and $M \subseteq X \times Y \times Z$, does there exist a subset $N \subseteq M$ and $|N| = m$, such that every two elements in N disagree in all three coordinates? This problem has been proved to be NP-Complete in [18].

The *Broadcast Time* problem that is proved to be NP-complete is the following. Given a graph $G = \{V, E\}$ with a specified set of vertices $V_0 \subseteq V$ and a positive integer k, does there exist a sequence $V_0, E_1, V_1, E_2, V_2, \ldots, E_k, V_k$, where $V_i \subseteq V, E_i \subseteq E (1 \leq i \leq k)$, $E_i = \{\{u, v\}, u \in V_{i-1}, v \notin V_{i-1}\}$, $V_i = V_{i-1} \cup \{v\}$, and $V_k = V$? Here, V_i is the set of informed vertices at round i, E_i is the set of message-passing edges at round i, and k is the total broadcast time. Let $|V_0| = 1$, then it is the exact case that broadcasting starts from an arbitrary single originator.

A 3DM problem can be reduced to a broadcast time problem with $k = 4$ in a certain graph G that is constructed from the set M in polynomial time m. Figure 2 shows the graph G. The independent sets V_0 and M have the same size and the bipartite subgraph induced by V_0 and M is complete. If $(x_i, y_j, z_k) \in M$, then the corresponding vertex of M in G is joined to the vertices x_i of X, y_j of Y and z_k of Z. For example, vertex (x_1, y_2, z_3) connects with vertices x_1, y_2 and z_3. All other edges are exactly as that shown in graph G.

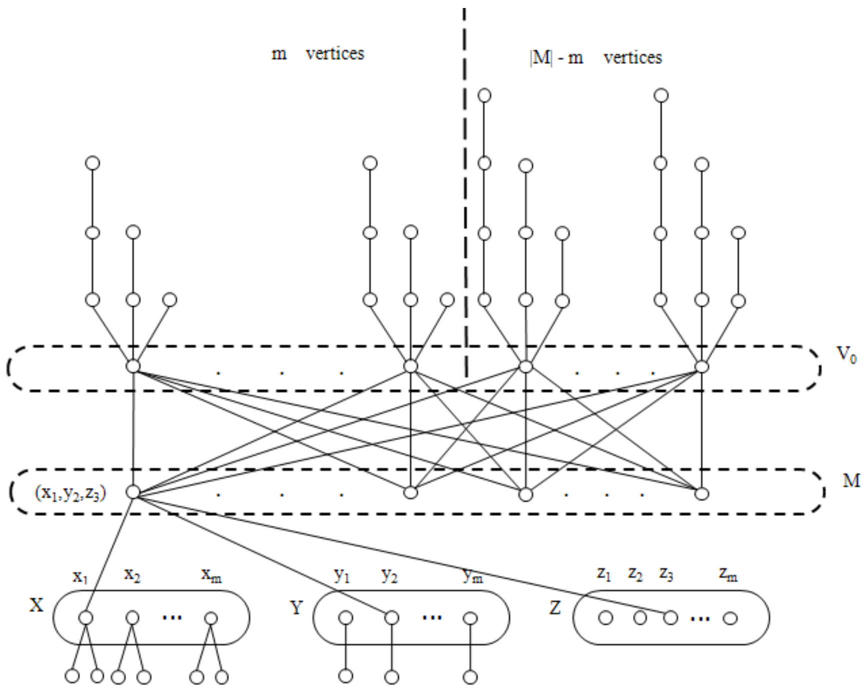


Figure 2: The graph G

Consider a solution for broadcast time problem in G, $|M| - m$ vertices of V_0 must pass the message in an upward direction in the first round, so that all vertices on the top line can be informed in round 4. Similarly, in order to inform X, Y, Z and all the vertices on the bottom line in round 4, the remaining m vertices in V_0 must send the message to an m-subset S of M in the first round. Thus, the vertices in S must be able to broadcast to 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.

Furthermore, we need 3DM to be reducible to the problem that determines broadcast time for an arbitrary graph G with an arbitrary originator u. Hence, a new graph H obtained from graph G will be used. Figure 3 demonstrates the graph H, which is constructed as follows. An independent vertex set $U = \{u_1, u_2, \ldots, u_m\}$, a vertex u and edges $\{(u, u_i), 1 \leq i \leq m = |V_0|\}$ are added to graph G. Every vertex u_i joins $m - i$ paths of length $6, 7, \ldots, m + 5 - i$. m edges are added to create a matching between U and V_0.

Consider a solution for the problem which determines whether $b(u) = m + 5$ in graph H. Each vertex u_i will be informed at round i, then will broadcast the message to the paths connected to it in the order of decreasing path length. Finally u_i informs the vertex in V_0 that is matched to it at round $m + 1$. Thus, every vertex in V_0 will be informed at round $m + 1$. Broadcasting in graph G will be done in 4 rounds, and $b(u) = m + 5$. The broadcast scheme in subgraph G is the solution for the broadcast time problem with $k = 4$. Hence, we can say that a broadcast time problem with $k = 4$ is equivalent to determining if $b(u) = m + 5$ in graph H.

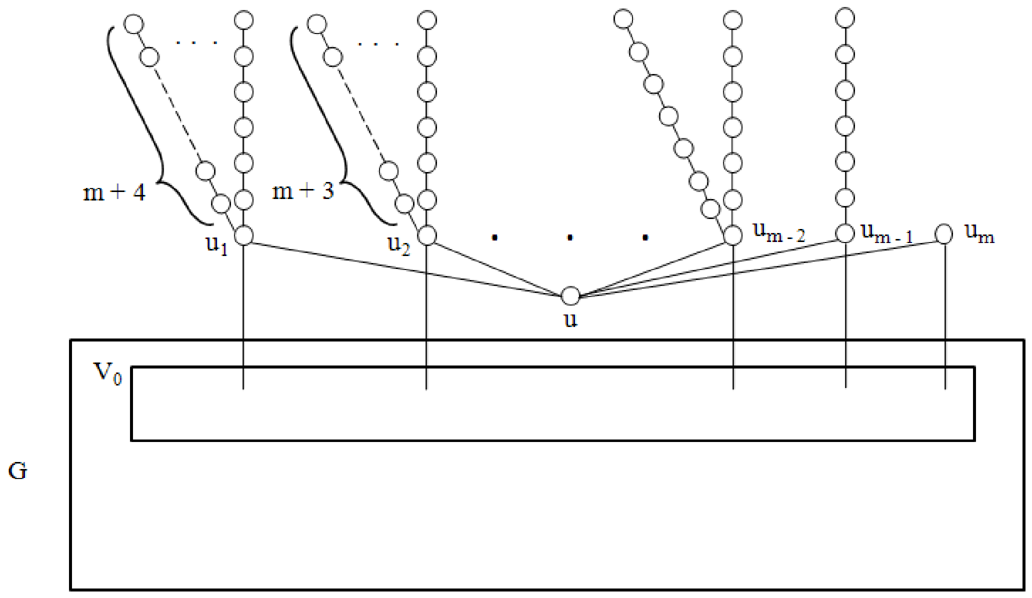


Figure 3: The graph H

We have seen that the broadcast time problem is NP-Complete for arbitrary graph. However, it is also NP-Complete for more restricted topologies, such as planar graphs [26, 27], and bounded degree graphs [5, 8, 33]. In addition, other results on the inapproximability of this problem are provided in [38].

1.3 Thesis Outline

This thesis is organized as follows. Chapter 1 gives an introduction of broadcasting in computer networks. In Chapter 2, some background knowledge, including commonly used topologies and broadcast heuristics, will be introduced. Moreover, the Minimum-Weight Cover (MWC) problem and the algorithm solving it are introduced in this chapter as well. In Chapter 3, two new heuristics, which pass the message via shortest paths, are formally described. In Chapter 4, simulation results concerning the heuristics are provided. Several commonly used topologies and four network models are involved in the simulations. Finally, Chapter 5 concludes this thesis and discusses possible future work.

2 Background

In this chapter, we will present a brief review of the commonly used interconnection topologies, broadcast heuristics with good performance in practice. We also present an algorithm solving Minimum-Weight Cover (MWC) problem and introduce a modified algorithm as well.

2.1 Commonly Used Topologies

In this section, several commonly used topologies are introduced. For more details the readers can refer to [14], [24], [30] and [21].

The Path P_n. A path is a graph of a sequence of vertices such that each vertex has an edge connecting it to the next vertex in the sequence. P_n has n vertices, a diameter that is equal to $n-1$, and maximum degree of 2. The broadcast time of P_n is equal to $n-1$. This value is achieved by the end vertices which have the maximum broadcast time in the path. Figure 4 shows a path with six vertices, where $b(P_6) = 5$.

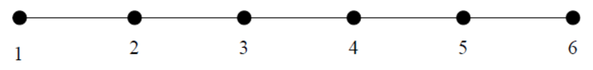


Figure 4: The example of path, $n = 6$.

The Cycle C_n. A cycle is a path such that the start vertex and end vertex are also connected by an edge. C_n has n vertices, the diameter is $\lfloor \frac{n}{2} \rfloor$, and maximum degree equals to 2. The broadcast time of C_n is equal to $\lceil \frac{n}{2} \rceil$. At each time unit, all the informed vertices pass the message to their uninformed neighbors. Figure 5 demonstrates two cycles with four and six vertices, and the broadcast time equals to 2 and 3 respectively.

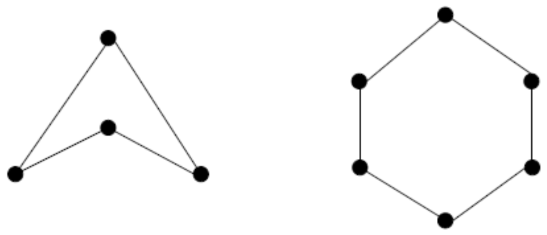


Figure 5: The examples of cycle graph, $n = 4$ and $n = 6$.

The Complete Graph K_n. A complete graph is a simple graph in which every pair of distinct vertices are connected by an edge. K_n has n nodes, diameter 1, and degree $n - 1$. It is easy to obtain that b(K_n) = $\lceil \log n \rceil$, since during each round every informed vertex can send message to an uninformed neighbor. Figure 6 shows two complete graphs of different dimensions, where $b(K_4) = 2$ and $b(K_6) = 3$.

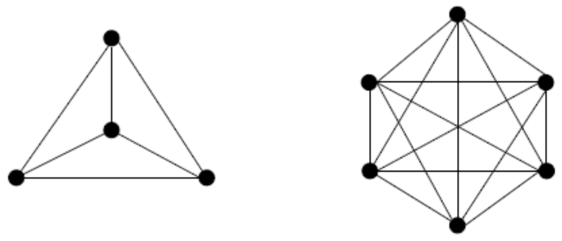


Figure 6: The examples of complete graph, $n = 4$ and $n = 6$.

The Hypercube H_m. The hypercube of dimension m, denoted by H_m, is the graph on 2^m vertices, where each vertex can be represented by a binary string of length m and is connected to those vertices whose binary string representation differ in exactly one position. H_m has 2^m nodes, $m2^{m-1}$ edges, diameter m and each vertex has exactly degree m. An

$(m+1)$-dimensional hypercube can be constructed by two m-dimensional hypercubes by just connecting each pair of the corresponding vertices. The broadcast time of hypercube is exactly equal to m, because at each round the number of informed vertices doubles. Figure 7 illustrates two hypecubes of dimension 3 and 4.

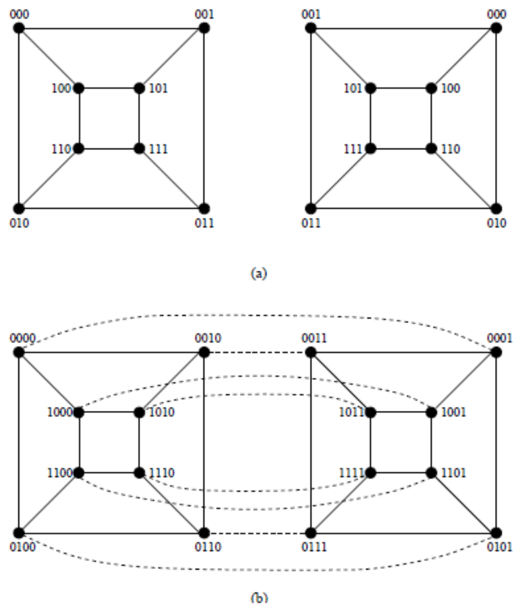


Figure 7: The example of hypercubes.

The Cube-Connected Cycles CCC_m. The CCC_m is a modification of the hypercube H_m by replacing each vertex of the hypercube with a cycle of m vertices. The i-th dimension edge incident to a node of the hypernode is then connected to the i-th node of the corresponding cycle of the CCC_m. Thus, CCC_m has $m2^m$ nodes, diameter equals to $2m + \lfloor\frac{m}{2}\rfloor - 2$, and the maximum degree is 3. [31] provides that $b(CCC_m) = \lceil\frac{5m}{2}\rceil - 1$, first every informed vertex sends the message to the hypercube neighbor, then to the right neighbor on the ring, and finally to the left one. Figure 8 shows a 3-dimensional cube connected cycle.

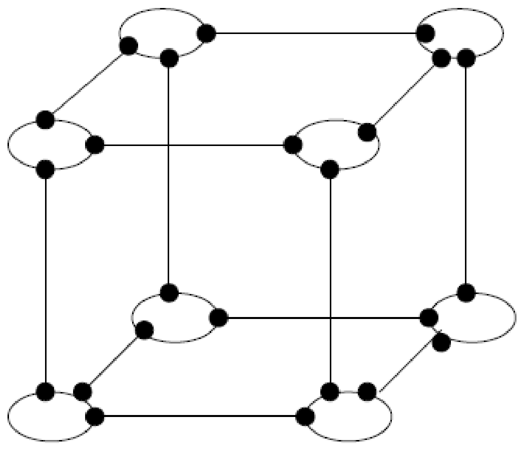


Figure 8: The example of Cube Connected Cycles, $m = 3$.

The Shuffle-Exchange SE_m. SE_m is the graph whose vertices can be represented by binary strings of length m. Each edge of SE_m connects vertex αa, where α is a binary string of length $m-1$ and a is in $\{0, 1\}$, with vertex αc and vertex $a\alpha$, where c is the binary complement of a. SE_m has 2^m vertices, diameter is $2m - 1$, and maximum degree 3. In [23], it is proved that $b(SE_m) \leq 2m - 1$. Figure 9 shows a Shuffle-Exchange graph of dimension 3.

The DeBruijn DB_m. DB_m is the graph whose nodes can be represented by binary strings of length m and whose edges connect each string $a\alpha$, where α is a binary string of length $m - 1$ and a is in {0,1}, with the strings αb, where b is a symbol in {0,1}. DB_m has 2^m vertices, diameter m and maximum degree 4. [28] provides the lower bound $b(DB_m) \geq 1.3171m$, and [6] proves the upper bound, $b(DB_m) \leq 1.5m + 1.5$. Figure 10 illustrates a DeBruijn graph of dimension 3.

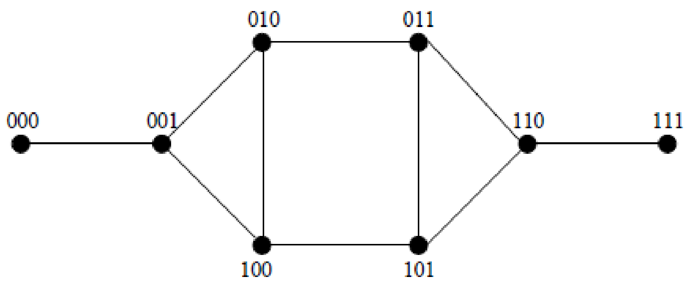


Figure 9: The example of Shuffle-Exchange graph, $m = 3$.

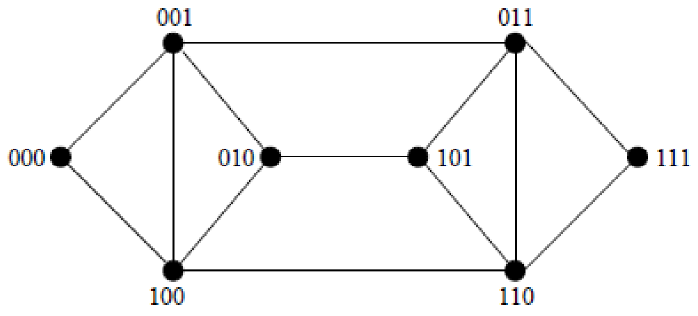


Figure 10: The example of DeBruijn graph, $m = 3$.

The Butterfly BF_m. BF_m has vertex-set $V_m = \{0, 1, \ldots, m-1\} \times \{0,1\}^m$, where $\{0,1\}^m$ denotes the set of length-m binary strings. For each vertex $v = \langle i, \alpha \rangle \in V_m, i \in \{0, 1, \ldots, m-1\}, \alpha \in \{0,1\}^m$, we call i the level and α the position-within-level of v. The edges of BF_m are of two types: For each $i \in \{0, 1, \ldots, m-1\}$ and each $\alpha = a_0 a_1 \ldots a_{m-1} \in \{0,1\}^m$, the vertex $\langle i, \alpha \rangle$ on level i of BF_m is connected

- by a straight-edge with vertex $\langle (i+1) \bmod m, \alpha \rangle$ and
- by a cross-edge with vertex $\langle (i+1) \bmod m, \alpha(i) \rangle$

on level $(i+1) \bmod m$. Again, $\alpha(i) = a_0 \ldots a_{i-1}c_ia_{i+1} \ldots a_0a_{m-1}$, where c_i denotes the binary complement of a_i. BF_m has $m2^m$ nodes, diameter $\lfloor \frac{3m}{2} \rfloor$ and maximum degree 4. From [28], we know that $1.7417m \leq b(BF_m) \leq 2m-1$. Figure 11 illustrates a 3-dimensional Butterfly graph.

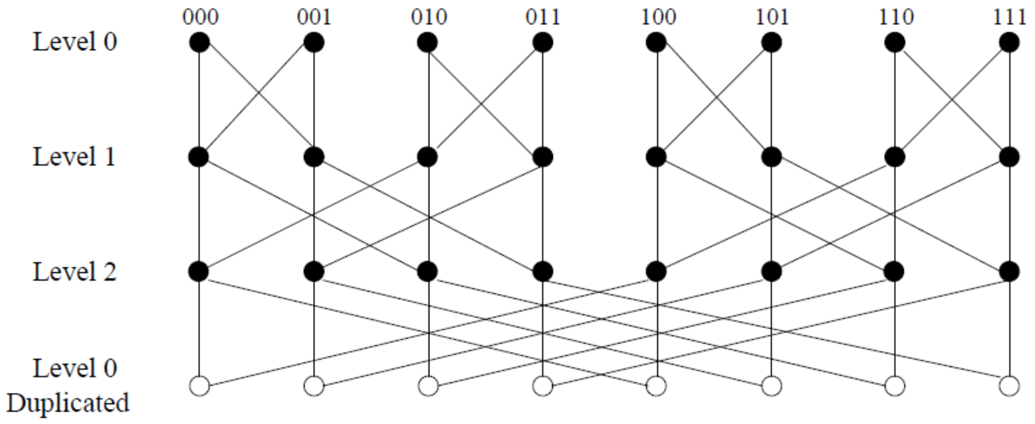


Figure 11: The example of Butterfly graph, $m = 3$.

The d-Grid $G[a_1 \times a_2 \times \ldots \times a_d]$. The d-dimensional grid (or mesh) is the graph whose nodes are all d-tuples of positive integers $(z_1, z_2, \ldots, z_d)$, where $0 \leq z_i < a_i$ for all i $(1 \leq i \leq d)$, and whose edges connect d-tuples which differ in exactly one coordinate by one. For example, in $G[3,3]$, vertex (1, 1) is connected to vertices $(0,1)$, $(2,1)$, $(1,0)$ and $(1,2)$. $G[a_1 \times a_2 \times \ldots \times a_d]$ has $a_1 \times a_2 \times \ldots \times a_d$ vertices, diameter $(a_1-1)+(a_2-1)+\ldots+(a_d-1)$, and maximum degree $2d$, if each a_i is at least three. [21] provides the broadcast time of a 2-grid, $b(G[a_1 \times a_2]) = a_1 + a_2 - 2$. Figure 12 shows a 2-Grid graph $G[4 \times 3]$.

The d-Torus $T[a_1 \times a_2 \times \ldots \times a_d]$. A d-Torus graph is a d-grid graph with both ends of rows and columns connected. Similarly, we use $T[a_1 \times a_2 \times \ldots \times a_d]$ to denote the d-Torus.

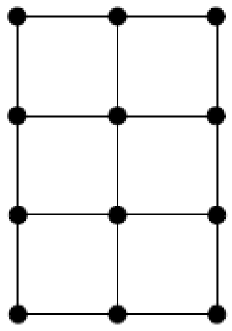


Figure 12: The example of 2-Grid graph with 12 vertices.

The optimal broadcast time of the 2-Torus graph is $\lceil\frac{a_1}{2}\rceil + \lceil\frac{a_2}{2}\rceil$, when a_1 or a_2 is even; and it is $\lceil\frac{a_1}{2}\rceil + \lceil\frac{a_2}{2}\rceil - 1$, when both a_1 and a_2 are odd [12]. The bounds on the broadcast time of the Torus are $D \leq b(T[a_1 \times a_2 \times \ldots \times a_d]) \leq D + \max(0, m-1)$, where $D = \sum_{i=1}^{d} a_i - d$, and m is the number of odd a_i. Figure 13 illustrates a 2-Torus graph $T[4 \times 3]$.

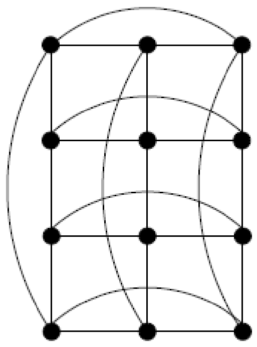


Figure 13: The example of 2-Torus graph with 12 vertices.

2.2 Previous Heuristics

The first algorithm that attempts to solve the minimum broadcast time problem is presented in [36] in 1981, and then another exact algorithm, based on dynamic programming, is designed by Scheuermann and Wu [37] in 1984. A backtracking algorithm for bounded degree networks is described in [19].

Since finding the minimum broadcast time of any originator in an arbitrary graph is NP-complete, many approximation algorithms and heuristics have been presented to determine the broadcast scheme with minimum time cost (see [2, 10, 17, 16, 29, 11, 15, 4, 35, 37, 13, 34, 39, 7]).

Given a graph $G = (V, E)$ and the originator u, the heuristic in [29] returns a broadcast scheme whose performance is at most $b(u, G)+Diam(u)+3\sqrt{|V|}$ rounds, where $Diam(u)$ is the diameter of u, and b(u, G) is the optimal broadcast time. Another well-known algorithm, presented in [35], is based on calculating the *poise* of a graph. The poise of a tree T is defined as the sum of the maximum degree of any vertex in the tree and the diameter of the tree. The poise of a graph G, denoted by $P(G)$, is defined as the minimum poise of any of its spanning trees. Computing the poise of an undirected graph is NP-hard. However, [35] present an $O(nm \log n)$-complexity heuristic to compute a spanning tree of a graph on n vertices and m edges, such that the poise of the tree is within $O(\log n)P(G) + O(\log^2 n)$. [35] also proves that $b(G) = O(P(G)\frac{\log n}{\log \log n})$. The time complexity of the algorithm is $O(nm \log^2 n)$, and the upper bound of the broadcast time is $O(\frac{\log^2 n}{\log \log n}b(G))$. Theoretically, the best upper bound is obtained by the algorithm presented in [11], which generates a broadcast scheme with $O(\frac{\log|V|}{\log\log|V|}b(G))$ rounds

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

In the following sections, two heuristics of value in practice are introduced in details. The Round-Heuristic [4] and the Tree Based Algorithm [20] are both outstanding heuristics, which have almost the same performance in most of commonly used topologies, and generate better performance in three network models from ns-2 simulator [1, 2, 9, 44]. In many variant topologies, their performances are very close to the optimal value or to the lower bound. Thus, they will be used to scale the new heuristics in this thesis. In the next two subsections, these two heuristics will be introduced.

2.2.1 Round_Heuristic

The Round_Heuristic is described in [4], which also presents the simulation results in several commonly used graphs. From its simulation results, we can say that its performance is quite close or equal to the optimal value. The Round_Heuristic is designed for both broadcasting and gossiping problems, and its broadcasting performance will be considered in this thesis.

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

Setting the weights rationally and effectively are the most significant steps. In [4], two different approachs are introduced to set the weight. One is called the *Potential Approach*, and the other is the *Breadth-First-Search*(BFS) approach. The potential approach assigns each edge(v, w) a weight equal to its poential, defined as the number of messages known by either v or w, but not by both of them. In broadcasting, the weight could only be 0 or 1.

Obviously, the potential approach is simple and requires little storage and runs very fast. However, as a pure local greedy algorithm, it lacks a global view. The BFS approach works much better in this aspect, although its cost is far more expensive. Before going to the details of BFS approach, several definitions should be presented. In a connected graph, the dispersion region $DR(p, t)$ of a message p is the set of vertices that know p at the beginning of round t. For any vertex $v, dist_v(p, t)$ denotes the shortest distance in the graph from v to a vertex $w \in DR(p, t)$. The set of border-crossing edges, denoted by $bce(p, t)$, is defined as $bce(p, t) = \{(v, w) \in E | v \in DR(p, t)$ and $w \notin DR(p, t)\}$. For any vertex $v \notin DR(p, t), bce_v(p, t)$ consists of all edges in $bce(p, t)$ that lie on the shortest path from $DR(p, t)$ to v. Figure 14 illustrates the dispersion region $DR(p, t)$ for a message p. The border-crossing edges, $bce(p, t)$, are drawn in bold. $dist_v(p, t) = 3$ and $bce_v(p, t) = \{e_1, e_2\}$.

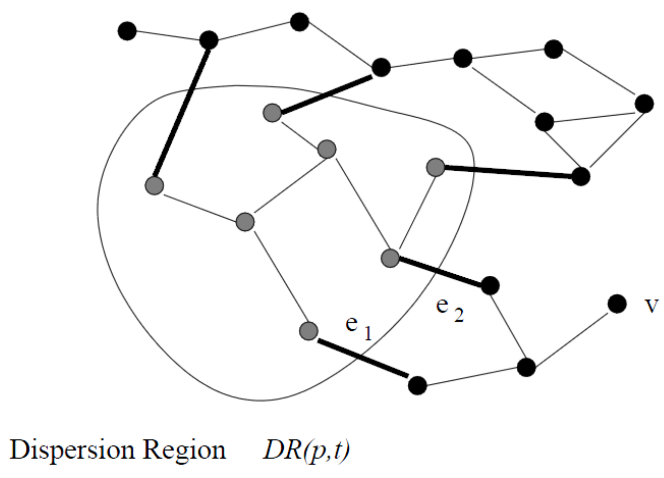


Figure 14: The dispersion region $DR(p, t)$ for some message p.

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

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

where $dist_v(p, t)$ and $bce_v(p, t)$ are calculated for every vertex v, at round t, and *Dist_Exp* and *Num_Exp* are two parameters. [4] applies a modified breadth first search algorithm, such that vertices are considered in order of increasing $dist_v(p, t)$. For vertex v with

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

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

The value of the weight depends heavily on the choice of the parameters, $Dist\_Exp$ and Num_Exp. Thus, the impact of the parameters play a significant role in the performance of the Round_Heuristic. Particularly, $Dist\_Exp$ is of great significance, which determines the influence of the distance between nodes and dispersion regions. Usually, values ranging from 0.25 to 60 are used. The precise choice for two topologies are mentioned in [4]: for the mesh graphs, $Dist\_Exp$ = 4, while for the butterfly graphs, $Dist\_Exp$ = 2.

In [4], simulation results of the heuristic in commonly used topologies are presented, including the Cube Connected Cycles, the Shuffle Exchange graphs, the Butterfly graphs as well as the deBruijn graphs. Many of these values are close or equal to the optimal broadcast time, some of which will be presented in Chapter 4.

2.2.2 Tree Based Algorithm

The Tree Based Algorithm (TBA) is presented in [20], whose general idea derives from the Round_Heuristic. In round t, TBA partitions the graph into two parts, the bright region and the dark region. The bright region is composed of all informed vertices, similar to the Dispersion Region in Round_Heuristic, while the dark region includes 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.

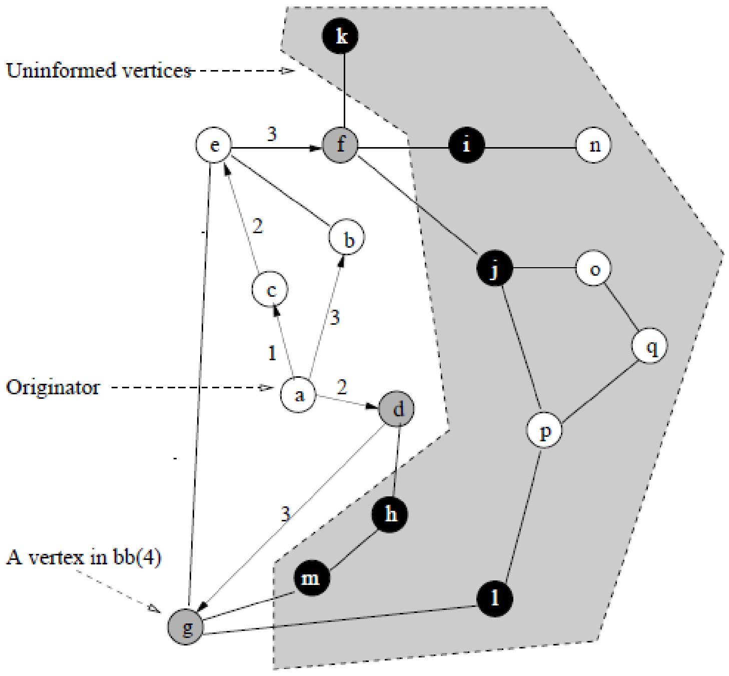


Figure 15: Definitions in TBA.

Figure 15 shows the definitions in TBA. Vertex a is the originator. After three rounds, all vertices in the shadowed area are still uninformed. The informed vertices 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. We can also say that vertices o, p and q are descendants of vertex j.

Same as the Round Heuristic, TBA applies maximum weighted matching to determine the message-passing edges between the bright region and the dark region. In each round, TBA performs a *breadth first search*(BFS) from $bb(t)$ towards all the uninformed vertices, and the parent-child relationship is determined by labeling each uninformed vertex v with $D(v,t)$. Then every uninformed vertex u will be assigned a weight, which is based on the strategy of the optimal broadcasting in trees. Let $w(u,t)$ stand for the weight of vertex u at round t. If u has no children, then $w(u,t) = 0$. Otherwise, $w(u,t) = \max\{w(C_i^u, t) + i\}$, where C_i^u is the i-th child of u, and without loss of generality, all the children of u are in decreasing order of their weights. After all the vertices in the dark region are assigned weights, TBA finds a maximum weighted matching between $bb(t)$ and their uninformed neighbors. A matching algorithm with time complexity $O(|E|)$ is applied by the heuristic. The matched edges are used to pass the message in that round.

The broadcast time is the round number during which all the vertices in the network are informed. Since each round takes time $O(|V| + |E|) = O(|E|)$, the total complexity is $O(R|E|)$.

[20] also provides simulation results for commonly used topologies as well as some network models, which show that in most cases TBA even has better results than the Round_Heuristic.

TBA has a refined version, which inherits the idea of choosing parameters from the Round_Heuristic, to obtain better broadcast time in some topologies. As a result, the weights are allowed to be decimal in the refinement.

2.3 The Minimum-Weight Cover Problem

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

Let $G(V_1, V_2, A, w)$ be a bipartite graph with bipartition (V_1, V_2), edge set A, and a weight function $w : A \mapsto Z^+$ on the edges, and no isolated vertices. Each vertex $v_1 \in V_1$ is called a *server*, and each vertex $v_2 \in V_2$ is called a *customer*. If a control function $F : V_2 \rightarrow V_1$, where $F(v_2) = v_1$ implies $(v_1, v_2) \in A$, and we say that v_1 controls v_2. For every server $v \in V_1$, the clients dominated by v is denoted by $D_1(v), \ldots, D_k(v)$, and the edges connecting v with its clients is denoted by $e_i^v = (v, D_i(v))$. Without loss of generality, all the clients dominated by v are in the order such that $w(e_i^v) \geq w(e_{i+1}^v)$ for $1 \leq i \leq k$.

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

A pseudopolynomial algorithm is given to solve the MWC problem. The basic idea is to check whether there exists a positive integer j, where $\min_e\{w(e)\}+1 \leq j \leq \min_e\{w(e)\}+$

$|V_2|$, and a control function F, such that $W(F) \leq j$. The algorithm constructs a flow graph G'_j based on G and j, with the property that G has a control function F with weight $W(F) \leq j$ iff it is possible to push $|V_2|$ units of flow from the source to the sink on G'_j. The details of algorithm are as follows.

MWC algorithm:

1. Set $j_1 = \min_e\{w(e)\} + 1$ and $j_2 = \max_e\{w(e)\} + |V_2|$.
2. Let $j = \lceil \frac{j_1+j_2}{2} \rceil$.
3. Construct the flow graph G'_j.
4. Calculate the maximum flow on G'_j from source to sink.
5. If the maximum flow is equal to $|V_2|$, then set $j_2 = j$, else set $j_1 = j$.
6. If j equals to j_2 and $j_2 \leq j_1 + 1$, then goto next step, else back to step 2.
7. Return the minimum control function F that corresponds to the maximum flow computed on G'_{j_1}.

The complexity of MWC algorithm mainly depends on which maximum flow algorithm is employed. Table 1 shows the different maximum flow algorithms and their time complexities. The Dinic's algorithm runs in $O(|E||\sqrt{V}|)$ time in networks with unit capacities.

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

Table 1: Maximum Flow algorithms.

Another crucial part of this algorithm is the construction of the flow graph G'_j. Create a source vertex s and a sink vertex t. Assume that w_v is the maximal weight that is less than or equal to $j-1$ of an edge incident to $v \in V_1$. Duplicate v into $w_v + 1$ different copies and arrange the copies in an arbitrary order $v_1, \dots, v_{w_v+1}$. 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. Figure 16 demonstrates an example of G'_3.

2.4 The MWC-Modified Algorithm

As known in previous section, for any bipartite graph $G(V_1, V_2, A, w)$ with bipartition (V_1, V_2), edge set A and weight function w, MWC algorithm finds the minimum value of $\max\limits_{v \in V_1}\{\max\limits_{i}\{i + w(e_i^v)\}\}$. Based on the control function obtained from the MWC algorithm, we make some modifications and try to make $\max\limits_{v_j \in V_1}\{\max\limits_{i}\{i + w(e_i^{v_j})\} + j\}$ as small

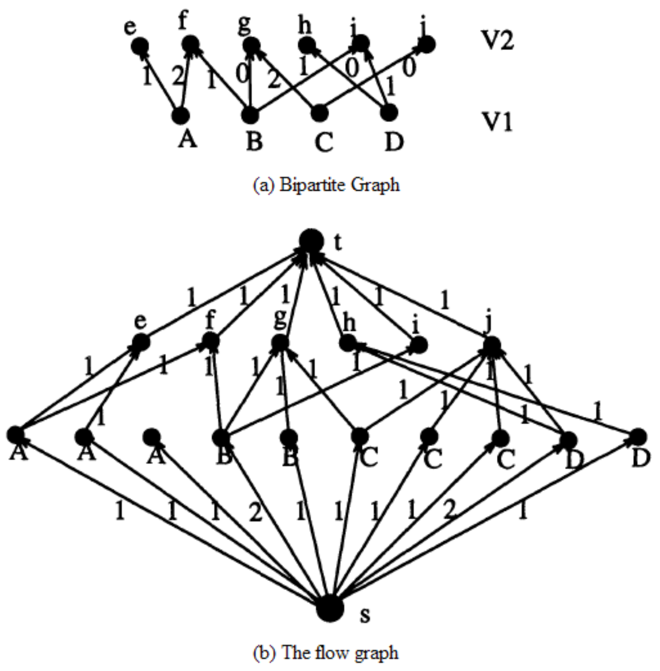


Figure 16: (a) A bipartite graph G. (b) Its corresponding flow graph G'_3.

as possible. We call this modified algorithm MWC-Modified.

MWC algorithm generates a control function $F : V_2 \rightarrow V_1$, and every server can be labeled with a weight $\max\limits_{e_i^{v_j} \in F}\{i + w(e_i^{v_j})\}$, denoted by w_S. Assuming that servers are in the descending order of their weights (i.e., $w_{S_j} \geq w_{S_{j-1}}$, S_j and S_{j-1} belong to $|V_1|$), we know that w_{S_1} is the minimum possible value. Now let MW start from $w_{S_1} + 1$ to $\max\limits_{F}\{w_{S_j} + j\}$, and try to generate a new control function F' such that $\max\limits_{F'}\{w_{S_j} + j\} \leq MW$, and keep MW as small as possible. The details of the algorithm are described as following.

MWC-Modified algorithm

Input: Bipartite graph $G(V_1, V_2, A, w)$, V_1 is the set of servers, V_2 is the set of customers

1. After performing MWC algorithm, obtain a control function F between servers and customers, and label every server with a weight $w_{S_j} = \max_{e_i^{v_j} \in F}\{i + w(e_i^{v_j})\}$.

2. Sort servers in the descending order of their weights, such that $w_{S_1} \geq w_{S_2} \geq \ldots \geq w_{S_{|V_1|}}$. max_weight denotes $\max_{F}\{w_{S_j} + j\}$.

3. Create buckets, and put all servers with weight w into bucket B_w. N_w is the sum of vertices whose weight is bigger than or equal to w.

4. Let MW start from $w_1 + 1$ to max_weight

   For each bucket from $B_{w_{S_1}}$ to $B_{w_{S_n}}$, if $N_w + w > MW$, then call procedure Re-Matching for each member in the bucket B_w, until $N_w + w \leq MW$.

   Repeat this step, until $N_w + w \leq MW$ for every bucket.

5. Update the control function and the weight of every server.

Procedure Re-Matching

Input: Server S and its matched customers $C_1, C_2, \ldots, C_n$, i.e., (S, C_i) is in the control function, where $1 \leq i \leq n$, and all customers are in the descending order of their weights.

1. For each customer C from C_1 to C_i (C_i: the first customer such that $C_i + i$ is equal to w_S), check every unmatched server S' of C, i.e., (S', C) is not in the control function. If $w_{S'} + N_{w_{S'}} < MW$, and $w_C + 1 \neq w_S$ or $w_C + 1 < w_{S'}$, then

```
            a) add $(S', C)$ to the control function, and remove $(S, C)$ from the control
            function,
            b) go to step 2.
        If $w_{S'} + N_{w_{S'}} >= MW$, and $w_C + 1 \neq w_S$ or $w_C + 1 < w_{S'}$,
            If $C$ will not increase the weight of $S'$,
            or if $w_C + 1 > w_{S'}$ and $N_{w_C+1} + w_C + 1 < MW$,
            or if $w_C + 1 \leq w_{S'}$ and $N_{w_{S'}+1} + w_{S'} + 1 < MW$, then
                a) add $(S', C)$ to the control function, and remove $(S, C)$ from the control
                function,
                b) go to step 2.

    2. Update weights and buckets of servers, and return.
```

In this thesis we apply Dinic's maximum flow algorithm, whose complexity is $O(|V|^2|E|)$, thus the total complexity of MWC algorithm is $O(|V|^2|E| \log |V|)$. The main part of MWC-Modified algorithm is step 4, where all servers are traversed to check if their weight can be reduced to meet $\max_{F'}\{w_{S_j}+j\} \leq MW$ by modifying the control function. The complexity of procedure Re-Matching is $O(\sum_{C_i \in V_2} deg(C_i)) = O(|E|)$, and thus the complexity of step 4 is $O(|V|^2|E|)$. The total complexity of MWC-Modified is $O(|V|^2|E| \log |V|)$, when applying Dinics maximum flow algorithm.

3 New Broadcast Heuristics

In this chapter, we will present two new heuristics, which broadcast messages from the originator to the rest of network via shortest paths.

3.1 Definitions

We start with some definitions which are essential to have a better understanding of the heuristics. All vertices and edges are supposed to be in a connected graph $G = (V, E)$.

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 $L(v) = L(u) + 1$.*

It's easy to find out that every two adjacent layers construct a bipartite graph. Figure 17 shows an simple example. Vertex c is the child of a, as well as the parent of d, e and f. Thus, all the vertices c, d, e and f are the descendants of a.

Definition 2 *If vertex u and vertex v are neighbors and $L(v) = L(u) + 1$, then v is called the child of u, and u is called the parent of v. Any child of vertex u is its descendant. Any of the children of the descendants of u is also a descendant of u.*

In figure 17, vertex c is the child of a as well as the parent of vertices d, e and f. Thus, all the vertices c, d, e and f are the descendants of a.

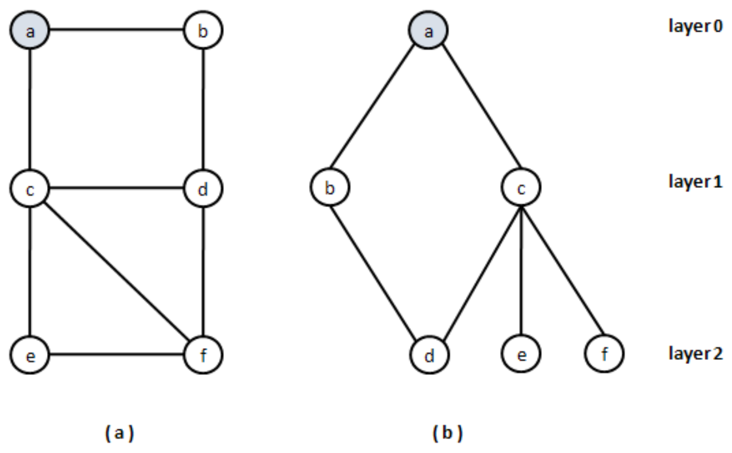


Figure 17: An example of layer graph. (a) The original graph G (b) The layer graph G_L

Definition 3 *In order to estimate the broadcast time of any vertex v in graph G, we employ the concept of estimated time, which is denoted by $EB(v)$. It can be calculated by the following recursion.*

1. $EB(v)$ is equal to 0, if vertex v has no children.
2. If v has k children, $c_1, c_2, \ldots, c_k$, 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 $1 \leq i \leq k$.

Here, we will introduce an algorithm to calculate $EB(v)$ if $EB(c_i)$ is given, where c_i is the child of v for $0 \leq i < k$. The complexity of this algorithm is $O(k)$.

1. Find $\max\{EB(c_i)\}$ of v, and denote it by MAX.
2. Create k buckets, and number them from 0 to $k - 1$.

3. Consider any child c of v, if $MAX - i \geq EB(c) > MAX - i - 1$, put c into the ith bucket. Only the minimal value and the number of elements are necessary to record. $SUM(i)$ denotes the number of elements in the first i buckets, and $MIN(i)$ denotes the minimal value in the ith bucket.

4. Finally, $EB(u) = \max\{SUM(i) + MIN(i)\}$, for $0 \leq i < k$.

The proof of the last step is as follows: Given a vertex v with k children, $c_1, c_2, \ldots, c_k$, which are ordered such that $EB(c_i) \geq EB(c_{i+1})$, then $EB(v) = \max\{EB(c_i) + i\}$, for $1 \leq i \leq k$. $EB(c_i) + i$ is called the *order-weight* of c_i. As $MAX - i \geq EB(c) > MAX - i - 1$ for any child c in the ith bucket, the maximum difference among all EBs in this bucket is less than 1. Therefore, in the ith bucket, the child with the minimum EB has the maximum *order-weight*, which is equal to $SUM(i) + MIN(i)$. Thus, $\max\{SUM(i) + MIN(i)\}$ is the maximum *order-weight* of all the children, which is $EB(v)$.

Proposition 1 *For any vertex v in a tree, $EB(v)$ is exactly the time that vertex v needs to broadcast the message to all of its descendants.*

Proof: Vertex v and its descendants make up of a tree rooted at v. If vertex v has n children, $v_1, v_2, \ldots, v_n$, without loss of generality we assume that $EB(v_1) \geq EB(v_2) \geq \ldots \geq EB(v_n)$. Thus the optimal broadcast scheme of vertex v is the one where it sends the message to child v_i at round i. Since each vertex has one and only one parent in a tree, $EB(v)$ equals to $\max\{EB(v_i) + i\}$ which gives only one possible value. Hence, it is easy to conclude that $EB(v)$ is the broadcast time of root v. Figure 18 shows a simple tree, and the root is also the originator. The $EB(v)$ of each vertex v is labeled on the figure, which is equal to the time to broadcast a message from v to all of its descendants .

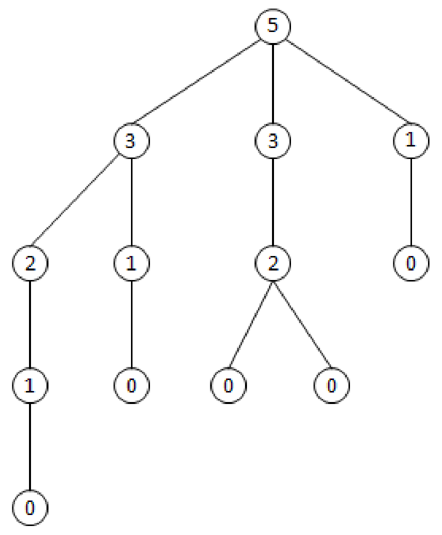


Figure 18: Broadcast time of Tree.

According to the definition of the layer graph, any two adjacent layers in a layer graph construct a bipartite graph. All the matching algorithms for bipartite graphs can be performed to generate a spanning tree. Based on the spanning tree, a broadcast scheme can be obtained. In particular, as known in Chapter 2, MWC algorithm is a matching algorithm that can be applied to generate a spanning tree.

Proposition 2 *The MWC algorithm is a 2-approximational algorithm for any layer graph with 3 layers.*

Proof: There is always one node, the originator, on the layer 0, and we assume that there are n nodes on the layer 1, v_1, v_2, $\ldots$,v_n. Based on the spanning tree generated by the

MWC algorithm in layer graph, without loss of generality, all the n nodes are sorted in the descending order of their weights, i.e., $WA_1 \geq WA_2 \geq \ldots \geq WA_n$, where WA_i denotes the weight of v_i on the layer 1, $1 \leq i \leq n$. Let $b(A, G)$ denote the broadcast time of the spanning tree. According to the Definition 3 and Proposition 1, $b(A, G) = \max\{i + WA_i\}$, $1 \leq i \leq n$, and we can conclude that $b(A, G) \leq WA_1 + n$.

There always exists an optimal broadcast scheme which generates the minimum broadcast time of a layer graph G. Based on the optimal broadcast scheme, similarly, all n nodes on the layer 1 are assumed to be in the order such that $WO_1 \geq WO_2 \geq \ldots \geq WO_n$, where WO_i is the weight of v_i on the layer 1 in optimal situation, $1 \leq i \leq n$. We denote the optimal broadcast time of the spanning tree of the layer graph G by $b(O, G)$. Since $b(O, G) = \max\{i + WO_i\}$, $1 \leq i \leq n$, we have $b(O, G) \geq \max\{WO_1 + 1, WO_n + n\}$. By the definition of the MWC problem, we know that WA_1 is the minimum possible value of v_1, hence $WO_1 \geq WA_1$. As $WO_1 \geq WA_1$ and $WO_n \geq 0$, we can conclude that $b(O, G) \geq \max\{WA_1 + 1, n\}$.

Now we have that $b(A, G) \leq WA_1 + n$ and $b(O, G) \geq \max\{WA_1 + 1, n\}$.

- If $WA_1 + 1 \geq n$, then $b(O, G) \geq WA_1 + 1$, thus $\frac{b(A,G)}{b(O,G)} \leq \frac{WA_1+n}{WA_1+1} \leq \frac{2n-1}{n} \leq 2$.
- If $WA_1 + 1 < n$, then $b(O, G) \geq n$, thus $\frac{b(A,G)}{b(O,G)} \leq \frac{WA_1+n}{n} \leq \frac{2n-1}{n} \leq 2$.

We can see that in any case $\frac{b(A,G)}{b(O,G)}$ is less than or equal to 2, thus the broadcast time generated by the MWC algorithm in any layer graph with 3 layer is 2-approximation.

3.2 Random Heuristic

In this section, a new heuristic called Random heuristic will be introduced. For every child who has several parents, the algorithm helps them randomly determine one parent and remove the edges with the other unmatched parents. Finally, each child is matched with only one parent. As a result a spanning tree of graph G is generated, in which every vertex connects with the root via the shortest path. This spanning tree can also generate the broadcast scheme of graph G. The complexity of Random heuristic is $O(|E|)$.

3.2.1 Algorithm

The details of the Random heuristic are as follows.

ALGORITHM Random

1. In any arbitrary graph G, start from the originator $o \in G$ and run the *breadth-first search* (BFS) algorithm. Remove all the edges that have not been traversed during BFS. As a result of this, a layer graph G_L is constructed.

2. For every two adjacent layers in G_L, randomly match a parent with each child who has more than one parent. Then remove all edges connecting the child with its unmatched parents. A spanning tree is created then.

3. In the spanning tree generated in the previous step, calculate $EB(v)$ for each vertex v.

4. $EB(o)$ is the broadcast time of o in graph G.

The main step of the Random heuristic is to randomly match each child with only one parent between adjacent layers in the layer graph, that is the reason why we call it Random heuristic. Figure 19 illustrates a simple example of the Random heuristic. The original graph G and its layer graph G_L are shown in Figure 19(a) and Figure 19(b). In G_L, we can see that vertex d has two parents, vertex b and vertex c. Hence there are two choices to match the vertex d with a parent. Actually in this case, d's choices have crucial influence on the final result. In Figure 19(c), we can see that if we pick b as vertex d's parent, and remove the edge connecting to vertex c, the broadcasting process from vertex a terminates in 3 rounds. Figure 19(d) shows the alternate choice, where vertex d chooses c as its parent and the broadcast time increases to 4.

3.2.2 Complexity

The Random heuristic adopts a simple random matching strategy in the bipartite graph so as to achieve a low complexity $O(|E|)$. The main part of the first step is the BFS, which is accomplished in $O(|V| + |E|) = O(|E|)$ time. Step 2 traverses all vertices and randomly matches children to parents. Thus the complexity is $O(|V|)$. The last step is to calculate $EB(v)$ for every vertex v. The complexity of calculating $EB(v_i)$ is $O(deg(v_i))$, where $deg(v_i)$ is the degree of v_i. Hence, the total complexity of step 3 is equal to $\sum_{i=1}^{n} O(deg(v_i)) = O(E)$, since $\sum_{i=1}^{n} deg(v_i) = 2|E|$. The complexity of Random heuristic equals to $O(|E|) + O(|V|) = O(|E|)$.

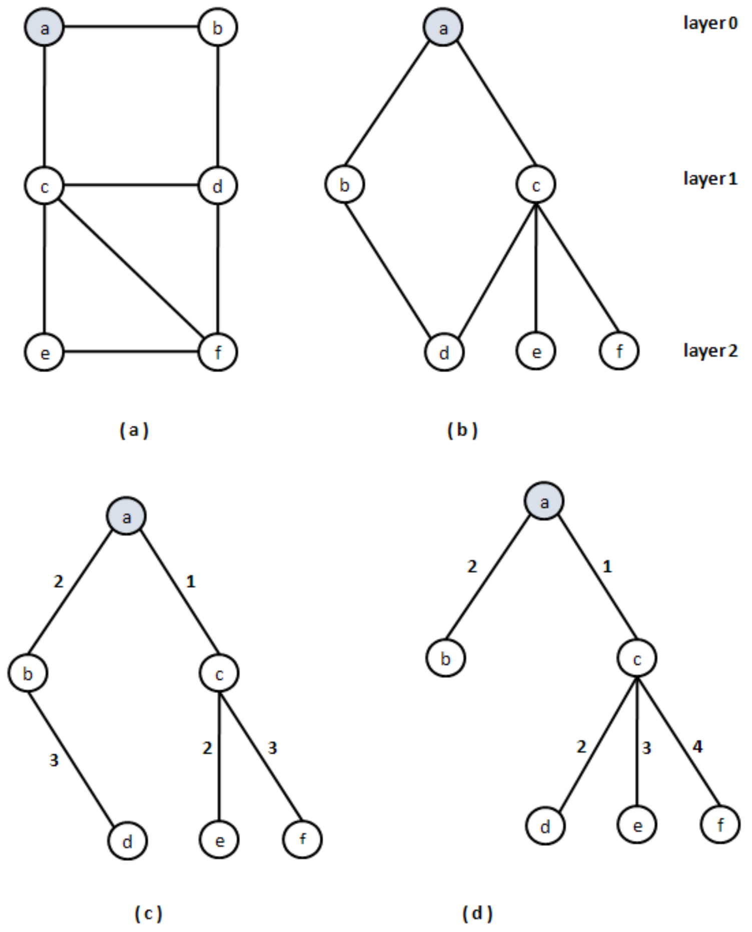


Figure 19: An example of Random Heuristic. (a) The original graph G with originator a. (b) The layer graph G_L. (c) One possible broadcast scheme, total broadcast time equals to 3. (d) The other possible broadcast scheme, the broadcast time is equal to 4.

3.3 Semi-Random Heuristic

The Random heuristic has a low time complexity, and generates a scheme where every member in the network receives the message via the shortest path. However, it makes random decisions when matching children and parents. Therefore, Semi-Random heuristic is designed to have some improvement over the random algorithm. Instead of matching randomly between adjacent layers, it employs a strategy that tries to make $\max\{EB(p_i)\}$ as small as possible, for each parent p_i on the same layer. The complexity of an algorithm is always a critical factor that has to be considered when designing any algorithm. The complexity of the Semi-Random heuristic is the same as that of the random one.

3.3.1 Algorithm

ALGORITHM Semi-Random

1. Start from the originator $o \in G$, run *breadth-first search* (BFS) algorithm. Remove all edges that have not been traversed during BFS. As a result, a layer graph G_L is constructed. Here, we assume there are k layers in G_L, and n_l vertices on the layer l, where $0 \leq l \leq k-1$.
2. Label each vertex on the layer $k-1$ with weight 0.
3. In the layer graph G_L, for each layer l starting from $k-2$ to 1, call procedure SRM.
4. Finally, $EB(o)$ is the broadcast time of originator o in graph G.

PROCEDURE SRM

Input: All vertices on layer l and layer $l+1$, and all edges between them.

1. On layer l, for each parent p_i starting from p_1 to p_{n_l},
   (a) pick all of p_i's neighbors with different weights on layer $l+1$ as its children, and remove all edges connecting these matched children with other parents on layer l,
   (b) label p_i with $EB(p_i)$ without considering its unmatched children.
2. For each unmatched child,
   (a) match it with a parent which has the smallest weight, and remove its edges connecting to other parents,
   (b) update the weight $EB(p)$ for its matched parent p.

According to the definition of estimated time, it is easy to conclude that children of identical weight connected to the same parent will increase the parent's estimated time. SRM is a greedy algorithm that tries to make each parent connect with children with different weights. An example is illustrated to help comprehend the algorithm. Figure 20 shows the original graph G with the originator a. Figure 21 shows the layer graph G_L with 5 layers, after breadth first search on G and removing all untraversed edges. The estimated time for each child on layer 4 is calculated and labeled on the graph.

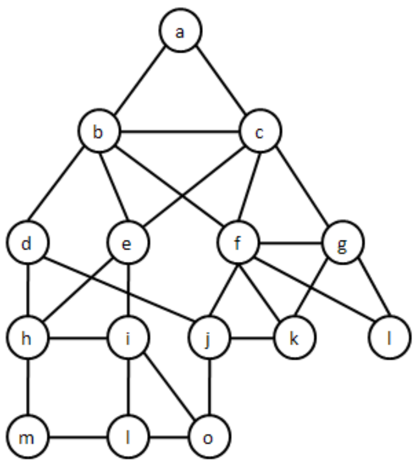


Figure 20: The original graph G with originator a.

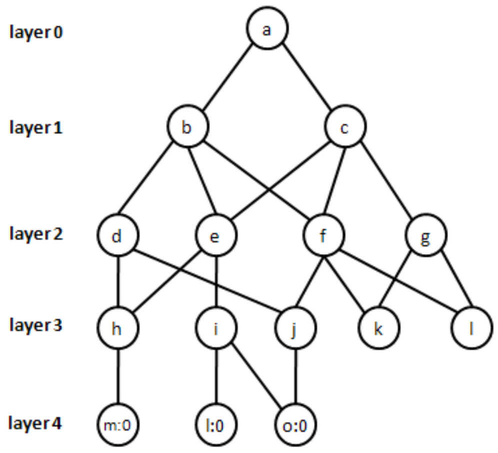


Figure 21: The layer graph G_L.

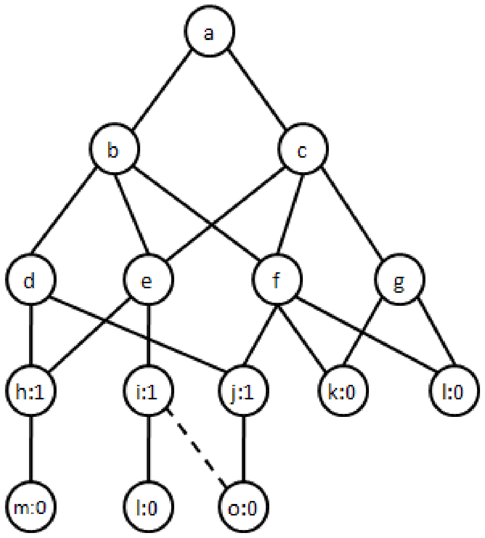


Figure 22: Perform SRM on layer 3 and layer 4.

Figure 22 shows how procedure SRM works between layer 3 and layer 4. Both vertex h and j have only one child, thus they have no choice and to be connected to vertices m and o respectively. Meanwhile, vertex o cuts off the edge with parent i. Hence, parent i has to take vertex l as its child. Based on the new matchings, the estimated time, EB for every vertex on layer 3 is calculated, and labeled on the graph. The dashed line on the figure represents a removed edge.

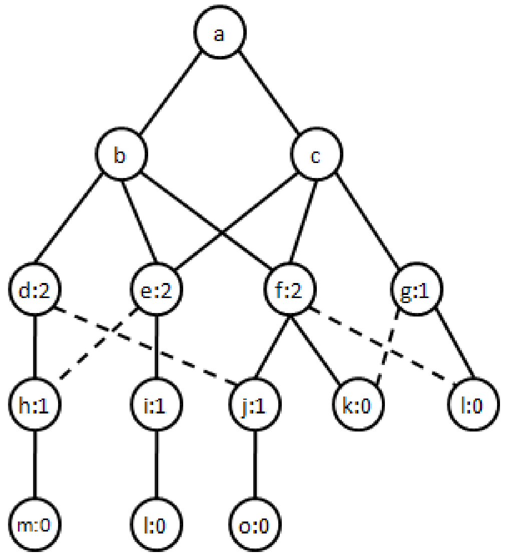


Figure 23: Perform SRM on layer 2 and layer 3.

Figure 23 illustrates the details of the matching between layer 2 and layer 3. First, parent d randomly chooses h as its child, since both h and j have the same weight. As a result, the edge between child h and parent e is removed, which is shown in a dashed line. Then parent e has to pick vertex i without any other choice. Parent f is able to adopt vertex j and vertex k, because they have different weights. Thus, both edges (d, j) and (g, k) are removed (shown in dashed lines). Finally, parent g has to take over child l. After the matching step, vertices d, e, f and g are labeled with weight 2, 2, 2, 1 respectively.

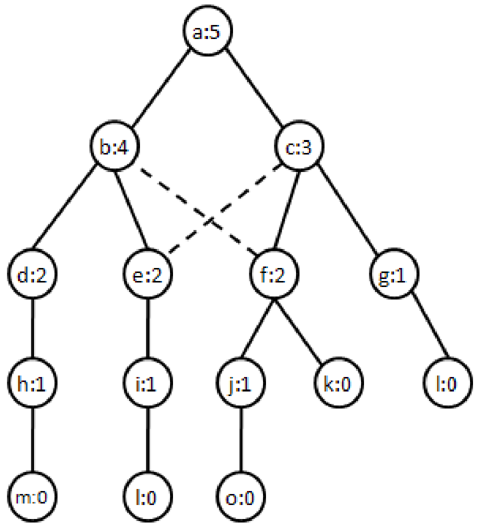


Figure 24: Perform SRM on layer 1 and layer 2.

Figure 24 shows the last step. First, parent b randomly chooses vertex d as its child, since all of its three children have the same weight. Here, $EB(b)$ equals to 3. Then parent c picks vertex f and vertex g, because they have different weights. Thus, $EB(c)$ equals 3 as well. Since now $EB(b) = EB(c)$, vertex e randomly chooses parent b, and $EB(b)$ is updated to 4. All edges removed during the matching are shown in dashed lines.

There is only one vertex on layer 0, the originator, thus it is easy to obtain that $EB(a) = 5$. We can see that a spanning tree is generated after matching, and a broadcast scheme can be obtained from this spanning tree. According to the Proposition 1, the broadcast time of the originator a is equal to $EB(a) = 5$.

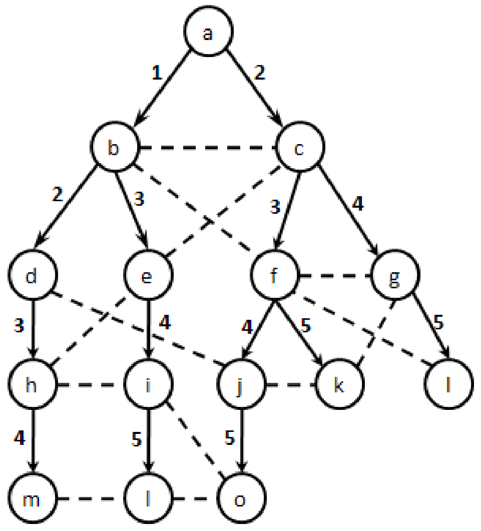


Figure 25: The broadcast scheme of graph G.

Figure 25 shows the broadcast scheme obtained from the spanning tree, by which each vertex receives the message through the shortest path. The arrows in the figure represents the broadcast scheme, while dashed lines denote all the edges not involved in the broadcasting.

3.3.2 Complexity

Even though there are more operations and steps compared to the Random heuristic, the Semi-Random heuristic does not increase the asymptotic time complexity. The breadth-first search and generation of the layer graph have a complexity of $O(|E|)$. The procedure SRM does not exceed this bound either. In the first step of SRM, each child has to determine if it has the same weight as others. Using binary tree to store the weights and applying binary search to check the weights, the total complexity for the whole graph is $O(\log 1) + O(\log 2) + \ldots + O(\log |V|) = O(\log |V|)$. Moreover, the calculation of $EB(p_i)$ takes $O(deg(p_i))$ for each parent, and the total complexity is $O(|E|)$ (already shown in Random heuristic). Hence, the complexity of the first step of SRM is $O(\log |V| + |E|) = O(|E|)$. In the second step of SRM, searching a parent with the smallest weight for each child c_i runs in $O(deg(c_i))$, thus the total complexity is $O(|E|)$. Then, updating $EB(v)$ for any single parent can be done in constant time, and the complexity of updating all parents is $O(|V|)$. We can conclude that the procedure SRM has a complexity of $O(|V|) + O(|E|) = O(|E|)$, and the total complexity of Semi-Random Heuristic is also $O(|E|)$.

4 Simulation Results and Comparisons with other Heuristics

In order to have a clear picture on the performance of the new heuristics in practice, we implemented and ran the heuristics in commonly used topologies and four network models from the NS-2 simulator. In this chapter all the simulation results will be presented and illustrated in tables and figures. In addition, all the heuristics mentioned in the previous chapters will be used for comparisons. In the first section, the simulation results in commonly used topologies are presented, and the next section provides the results in four network models from NS-2 simulator .

Before going into the details, the abbreviations used in this chapter are:

- RH: The result of Round_Heuristic from [4];
- TBA: The simulation result of Tree Based Algorithm obtained from [20];
- P-R: The simulation results of Random algorithm;
- S-R: The simulation result of Semi-Random algorithm;
- MWC: Like the SRM procedure in the Semi-Random heuristic, MWC algorithm (described in chapter two) can be performed between adjacent layers in the layer graph as well. Similarly, a spanning tree will be generated, and based on that we can obtain the broadcast scheme. Here, we inplement this algorithm and present its simulation results;

- MWC-M: Perform MWC-Modified algorithm (presented in chapter two) between adjacent layers in the layer graph, and generate a broadcast scheme.
- OPT: The optimal values;
- LOW: The lower bound;
- UP: The upper bound;
- D: The dimension number;
- 99%CI: 99% Confidence Interval;

4.1 Simulation Results and Comparisons in Commonly Used Topologies

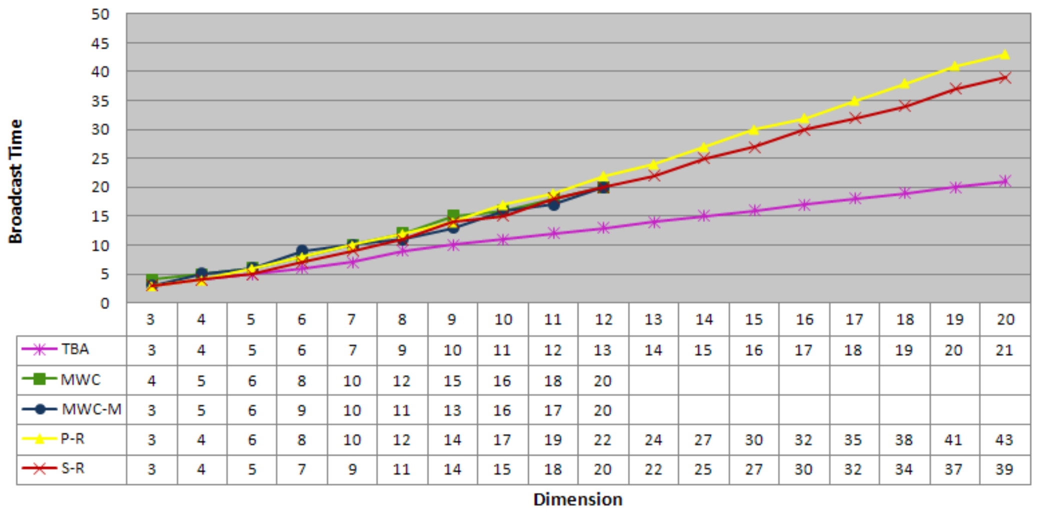
In this section, five commonly used topologies are studied, including Hypercubes, Cube Connected Cycles, deBruijn graphs, Shuffle-Exchange graphs, and Butterfly graphs as well. All the results and comparisons are illustrated in tables and figures.

4.1.1 Simulation in Hypercube

We already know that the broadcast time of hypercube is exactly equal to m, where m is the dimension. Table 2 shows the simulation results in Hypercubes of different dimensions, where the OPT values come from [24]. The Random and Semi-Random algorithms provide optimal broadcast time only when $m \leq 5$, otherwise with every increase of 1 in the dimension, the broadcast time increases by 2 or 3. We can also see that Semi-Random algorithm always has better results than the Random one. Compared with the results of the Round_Heuristic and the Tree Based Algorithm, the new algorithms do not perform well in Hypercubes. The Tree Based Algorithm has the best performance, whose broadcast time is equal to or just 1 more than the optimal value. When the dimension is large, the results of the new algorithms are about twice that of the Tree Based Algorithm. MWC and MWC-Modified algorithms almost have the same results as the new heuristics, nevertheless they have a bigger time complexity and can only work for Hypercubes of small dimension. We can conclude that the new algorithms are not suitable for Hypercubes. Figure 26 summarizes the results.

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

Table 2: Simulation results of different heuristics in 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

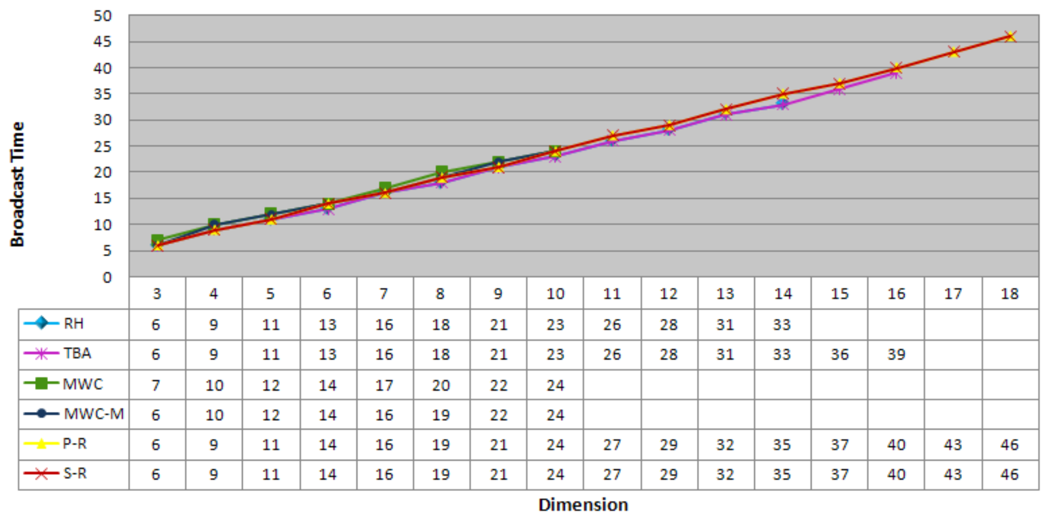
Figure 26: Simulation results in Hypercubes.

4.1.2 Simulation in Cube Connected Cycle

Table 3 shows the simulation results of the five algorithms in Cube Connected Cycles, where the LOW and UP values come from [24]. The Random and Semi-Random algorithms have the same broadcast time except when $D = 14$, and the results do not exceed the upper bounds until $D = 16$. When comparing the results with the other algorithms, we can find that the results of the new algorithms are equal to or just 1 more than that of Round_Heuristic and Tree Based Algorithm, which are both known as the best heuristics in practice. MWC and MWC-Modified algorithm provide similar results, and MWC-Modified algorithm works better when dimension is equal to 3, 7 and 8. Since the new algorithms have lower complexities, it is possible to calculate the broadcast time for Cube Connected Cycles of dimension 17 and 18, which would be harder with other heuristics. Figure 27 illustrates all these results, and we can see that all these heuristics almost have the same performance in Cube Connected Cycles.

D	LOW	UP	RH	TBA	MWC	MWC-M	P-R	S-R
3	6	7	6	6	7	6	6	6
4	9	9	9	9	10	10	9	9
5	11	12	11	11	12	12	11	11
6	13	14	13	13	14	14	14	14
7	16	17	16	16	17	16	16	16
8	18	19	18	18	20	19	19	19
9	21	22	21	21	22	22	21	21
10	23	24	23	23	24	24	24	24
11	26	27	26	26	-	-	27	27
12	28	29	28	28	-	-	29	29
13	31	32	31	31	-	-	32	32
14	33	34	33	33	-	-	35	34
15	36	37	-	36	-	-	37	37
16	38	39	-	39	-	-	40	40
17	-	-	-	-	-	-	43	43
18	-	-	-	-	-	-	46	46

Table 3: Simulation results of different heuristics in Cube Connected Cycles.

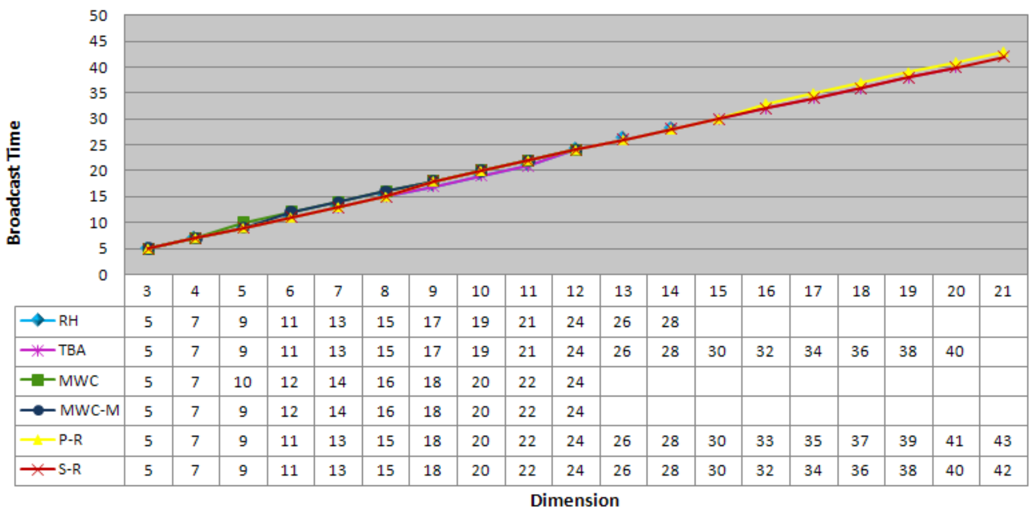


	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
RH	6	9	11	13	16	18	21	23	26	28	31	33				
TBA	6	9	11	13	16	18	21	23	26	28	31	33	36	39		
MWC	7	10	12	14	17	20	22	24								
MWC-M	6	10	12	14	16	19	22	24								
P-R	6	9	11	14	16	19	21	24	27	29	32	35	37	40	43	46
S-R	6	9	11	14	16	19	21	24	27	29	32	35	37	40	43	46

Figure 27: Simulation results in Cube Connected Cycles.

4.1.3 Simulation in Shuffle-Exchange Graph

The results in Shuffle-Exchange graphs are shown in Table 4 and Figure 28. The OPT values come from [24]. When $D \leq 15$, the Random algorithm has exactly the same performance as the Semi-Random one, while after that it results are always 1 more than that of the Semi-Random's. Both of them give the optimal value when $D \leq 8$. After that, the simulation results of the Semi-Random algorithm is always 1 more than the optimal value, while the Random one becomes at most 2 more from the optimal value. Compared with the Round_Heuristic and Tree Based Algorithm, the Semi-Random algorithm has the same performance except when the dimension is equal to 9, 10 and 11. Moreover, the Semi-Random algorithm also generates the new upper bound for the Shuffle-Exchange graph of dimension 21. The MWC-Modified algorithm performs better than the MWC algorithm only when the dimension is 5. Figure 28 shows that all these heuristics have similar performance in Shuffle-Exchange graphs.



	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
◆ 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	43
✕ S-R	5	7	9	11	13	15	18	20	22	24	26	28	30	32	34	36	38	40	42

Figure 28: Simulation results in Shuffle-Exchange graphs.

D	OPT	RH	TBA	MWC	MWC-M	P-R	S-R
3	5	5	5	5	5	5	5
4	7	7	7	7	7	7	7
5	9	9	9	10	9	9	9
6	11	11	11	12	12	11	11
7	13	13	13	14	14	13	13
8	15	15	15	16	16	15	15
9	17	17	17	18	18	18	18
10	19	19	19	20	20	20	20
11	21	21	21	22	22	22	22
12	23	24	24	24	24	24	24
13	25	26	26	-	-	26	26
14	27	28	28	-	-	28	28
15	29	-	30	-	-	30	30
16	31	-	32	-	-	33	32
17	33	-	34	-	-	35	34
18	35	-	36	-	-	37	36
19	37	-	38	-	-	39	38
20	39	-	40	-	-	41	40
21	-	-	-	-	-	43	42

Table 4: Simulation results of different heuristics in Shuffle-Exchange graphs.

4.1.4 Simulation in DeBruijn Graph

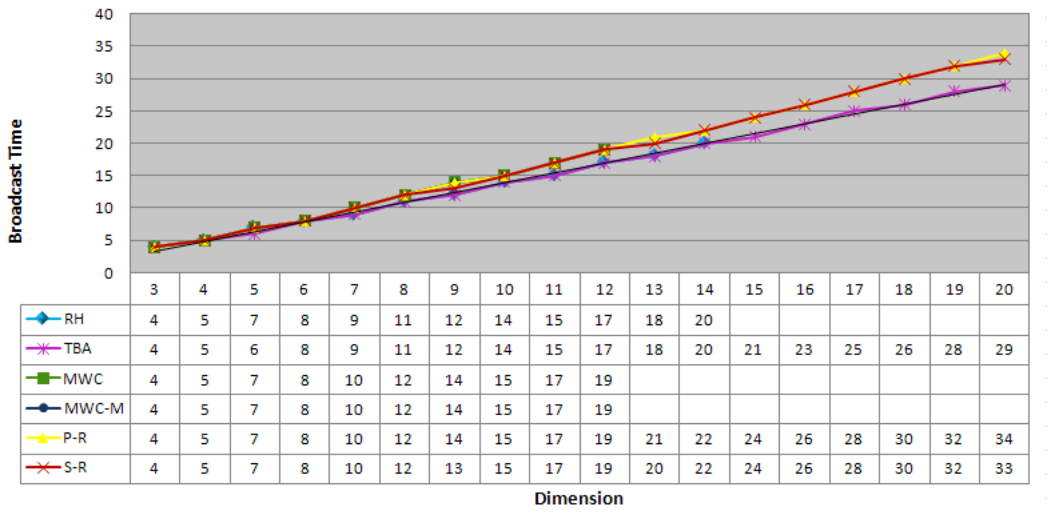
The lower bounds of DeBruijn graphs are calculated with the formulas in [28], thus they just hold asymptotically. That is why we can find some simulation results which are less than the lower bounds, such as the broadcast time of dimension 4 shown in table 5.

In table 5 we can see that the simulation results of different heuristics in DeBruijn graphs. The LOW and UP values come from [24]. In most cases, the Random algorithm generates the same results as the Semi-Random one, with the exception when the dimension is equal to 9, 13 and 20. Moreover, their results do not exceed the upper bound until dimension is

17, and after that each result is just one or two rounds over the upper bound of the broadcast time. Again, the Round_Heuristic and Tree Based Algorithm have almost the same results, which are equal or close to the lower bounds. When $D \leq 14$, the results of the new heuristics are at most 2 more than that of the Round_Heuristic and Tree Based Algorithm, and after that the difference increases to 3 and 4. The MWC and MWC-Modified algorithms have exactly the same simulation results, which are also similar to the new heuristics. In Figure 29, we can see that the Round_Heuristic and the Tree Based Algorithms have better results.

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

Table 5: Simulation results of different heuristics in DeBruijn graphs.



	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

Figure 29: Simulation results in DeBruijn graphs.

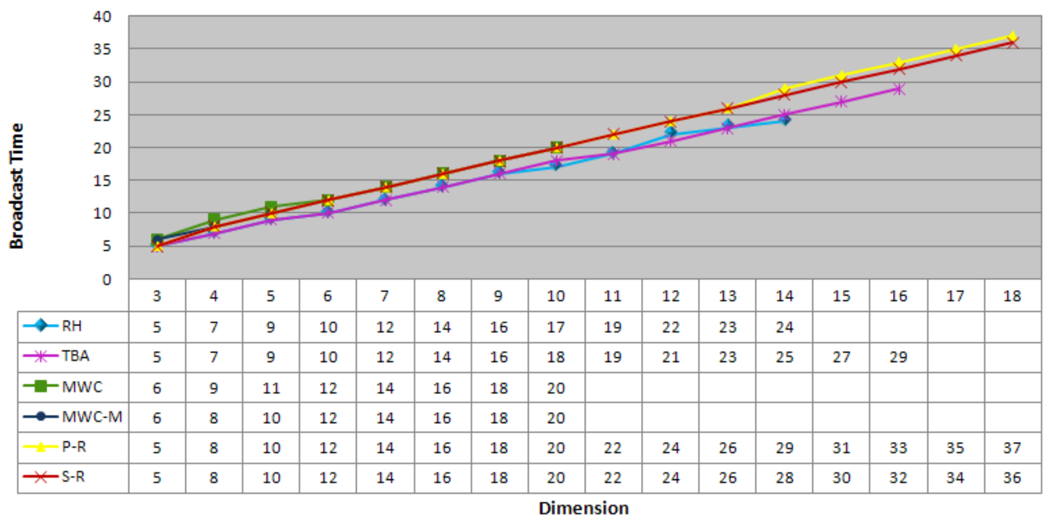
4.1.5 Simulation in Butterfly Graph

Observing table 6, we can find that the new heuristics have the same results when $D \leq 13$, and after that the Semi-Random algorithm performs better than the Random one. However, most results of the new heuristics are one or two rounds over the upper bounds of Butterfly graphs. Because of the low complexity of the heuristics, we are able to calculate the broadcast time for Butterfly graphs of dimension 17 and 18. On the other side, the Round_Heuristic works the best in the comparison, and the Tree Based Algorithm has the same performance except when dimension is 10 or 14. When increasing the dimension, the difference between the results of the new heuristics and of the previous heuristics also rises from 1 to 4. Again, the MWC and MWC-Modified algorithm have similar results to that of the Random and Semi-Random algorithms, however they only work in small graphs . We can conclude that the new heuristics are not the best candidates for the Butterfly graphs. Figure 30 shows the difference of theses heuristics. The LOW and UP values come from

[24].

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

Table 6: Simulation results of different heuristics in Butterfly graphs.



	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
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	35	37
S-R	5	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36

Figure 30: Simulation results in Butterfly graphs.

4.2 Simulation Results and Comparisons in NS-2 Models

In this section, all the simulation results collected in different network models are listed in tables. Four models are involved in the simulation, which will be introduced next. Since there is no optimal value for these network models, we just compare the results of our heuristics to those of previously known heuristics. Six heuristics will be compared to each other, which are the Round_Heuristic, the Tree Based Heuristic, the Random heuristic, the Semi-Random heuristic, and the heuristics applying MWC and MWC-Modified algorithms.

4.2.1 Introduction to Network Models

First, it is necessary to introduce the NS-2 simulator and the different network models. NS (Network Simulator) is developed for research in interconnection networks technology, which began as a variant of the REAL network simulator in 1989. NS-2 is the second generation of the simulator, and it can generate topologies automatically by using several models. In this section, four different network models are introduced, which are GT-ITM Random [44], GT-ITM Transit-Stub [44], Tiers [9], and BRITE Top-down Hierarchical models [32].

GT-ITM Random and Transit-Stub are both parts of Georgia Tech Internetwork Topology Models. The Pure-Random model is a standard random graph model, which is the prototype of many other models. This model places vertices in a plane randomly, and adds an edge between each pair of vertices with a probability p. This model is often considered in network researches, even though it does not correspond to any real network. When applying GT-ITM graph generator to generate pure random model, the probability p of an edge is the most crucial parameter to configure.

Transit-Stub is a well-known model for the Internet. The Internet can be regarded as a set of routing domains, a group of hosts on the Internet, can be considered to be an independent network. All vertices in a domain share the same routing information. There are two kinds of routing domains, one of which is the stub domain while the other is called the transit domain. A stub domain carries only traffic that originates or terminates in the domain. Meanwhile, a transit domain never has this restriction. The function of transit domains is to interconnect stub domains efficiently. For instance, stub domains generally correspond to some interconnected LANs (Local Area Networks) such as campus networks, while transit domains are usually regarded as MANs (Wide Area Networks) or WANs (Metropolitan Area Networks).

A transit domain is made up of a set of backbone nodes. In a transit domain, each backbone node may either connect to other transit domains, or connect to a number of stub domains, via gateway nodes in the stubs. Stub domains can be classified as single-homed or multi-homed stub: single-homed stubs connect to only one transit domain, while multi-homed ones connect to more than one transit domain. Moreover, some stub domains may have connections to other stubs, and transit domains can also be organized in hierarchies. Figure 31 shows an example of internet domain structure.

Similar to the real Internet, GT-ITM Transit-Stub generates graphs containing intercon-

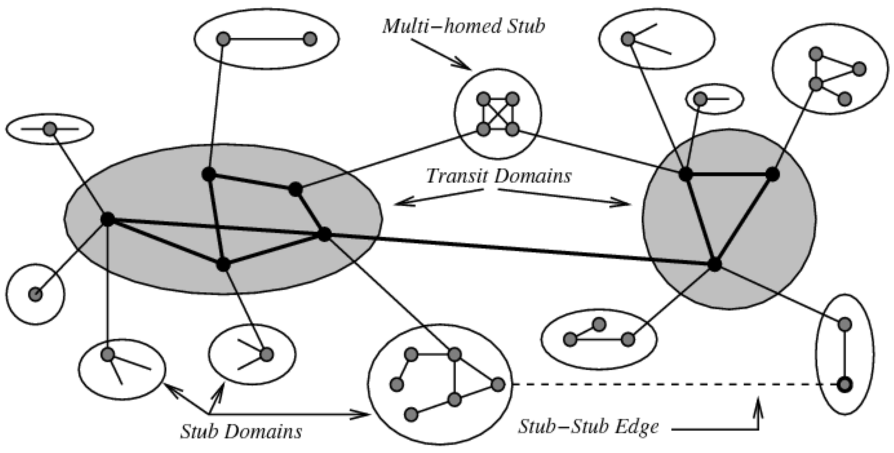


Figure 31: Example of Internet Domain Structure.

nected transit and stub domains. Furthermore, it can produce graphs having realistic average node degrees. The following parameters are available when applying GT-ITM Transit-Stub model.

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

The third model, the Tiers model, is one of the best network design models for networks research, and could accurately model the real network. This model generates topologies corresponding to the data communication networks such as IP network and ATM network.

Like the real networks, the Tiers model has a hierarchical structure, that is why it is called Tiers. The three levels of hierarchy are referred to as WAN, MAN and LAN levels, corresponding to the transit domains, stub domains and LANs respectively. The model does not currently support multiple WANs, thus the total number of WAN, is equal to 1. LANs such as Ethernet and Token Rings are basically modeled as star topologies. However, LANs should be modeled as being interconnected with small degree. Meanwhile, each stub domain (MAN) has a small number of connections to a transit domain (WAN). Figure 32 and 33 provide two Tiers networks of different scales.

The graph generator TIERS, created by Matthew B. Doar, is used to generate random networks. The major parameters chosen for this model are:

- N_W, the number of WANs; and S_W, the number of nodes in a WAN. As mentioned in last paragraph, N_W is always set to 1.
- N_M, the number of MANs; and S_M, the number of nodes per MAN.
- N_L, the number of LANs per MAN; and S_L, the number of nodes in each LAN.

The total number of nodes in the graph is $S_W + N_M S_M + N_M N_L S_L$. The other parameters of the model are:

- R_W,R_M and R_L; the degree of intra-networking redundancy in the WAN, MAN and LAN, respectively. This is expressed simply as the degree from a node to other nodes of the same type.

- R_{MW} and R_{LM}; 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}).

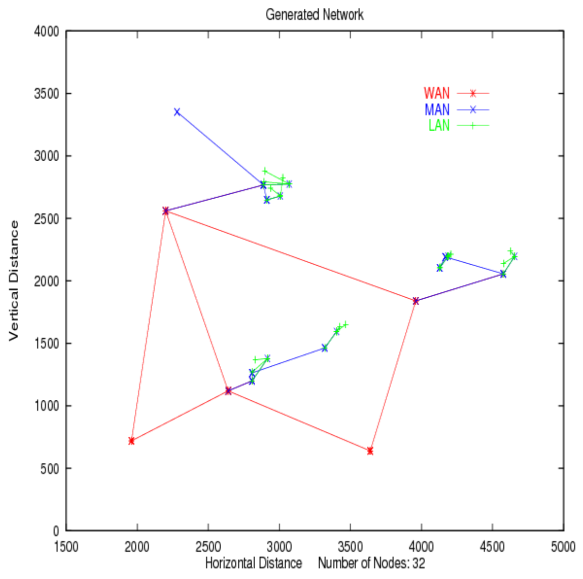


Figure 32: A Typical Tiers Internetwork.

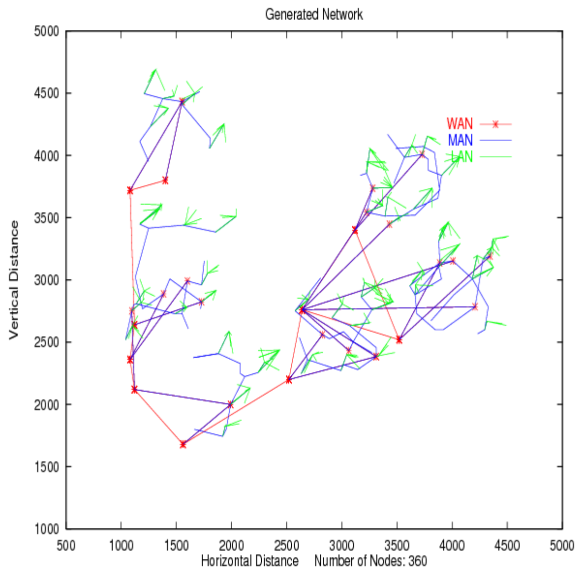


Figure 33: A Large Tiers Network.

Last, we introduce models from BRITE topology generator. BRITE (Boston university Representative Internet Topology gEnerator) is a universal topology generator, which generates synthetic topologies that accurately reflect many aspects of the actual Internet topology (e.g. hierarchical structure, degree distribution, etc.); combines the strengths of as many generation models as possible in a single generation tool; provides interfaces to widely-used simulation applications such as ns, SSF and OmNet++ as well as visualization applications.

Besides imported models, BRITE can generate different kinds of models, e.g. Flat Router-level Models, Flat AS-level Models, and Top-down Hierarchical Models.

Flat Router-level Models: This model represents router-level topologies. First, BRITE places the nodes on the plane randomly or in heavy-tailed way. Then, nodes are interconnected in one of two ways:

- Waxman's probability model: the probability of interconnecting two nodes u and v is given by $P(u, v) = \alpha e^{d/(\beta L)}$, where $0 < \alpha, \beta \leq 1$, d is the Euclidean distance from node u to node v, and L is t he maximum distance between any two nodes;
- Barabasi-Albert(BA) model: when a node i joins the network, the probability that it connects a node j already belonging to the network is given by $P(i, j) = \frac{d_j}{\sum_{k \in V} d_k}$, where d_j is the degree of target node j, V is the set of nodes that have joined the network and $\sum_{k \in V} d_k$ is the sum of outdegrees of all nodes that previously joined the network.

Flat AS-level Models: AS models represents AS-level topologies. This kind of models is very similar to Flat Router-level Models, except that AS nodes in the plane have the capability of containing associated topologies. Similarly, AS Waxman and AS BA models are provided in BRITE.

Top-down Hierarchical Models: Top-down models generate two-level hierarchical topologies. First, BRITE generates an AS-level topology according to one of the available flat AS-level models (e.g. Waxman or BA, etc.). Next, for each node in the AS-level topology, BRITE generates a router-level topology using a different generation model from the available flat models that can be used at the router-level. Then, BRITE uses one of four edge connection mechanisms (Random, Smallest degree, Smallest degree non-leaf, and Smallest k-degree) to interconnect router-level topologies as dictated by the connectivity of the AS-level topology. The final topology is obtained by flattening the hierarchical topology into a router-level topology composed of the individual topologies associated with each node at the AS-level. Figure 34 shows the three steps, labeled (1)-(3).

The basic edge connection methods provided with BRITE operate as follows. If (i, j) is a link in the AS-level topology, then pick a node u from the router-level topology associated with AS node i, $RT(i)$, and a node v from the router-level topology associated with the AS node j, $RT(j)$, such that:

- Random: u is picked randomly from $RT(i)$ and v randomly from $RT(j)$;
- Smallest degree: u and v are nodes with the smallest degrees in $RT(i)$ and $RT(j)$, respectively;

- Smallest degree non-leaf: u and v are nodes of smallest degree in $RT(i)$ and $RT(j)$ respectively but are not leaves;
- Smallest k-degree: u and v are nodes of degree greater that or equal to k in $RT(i)$ and $RT(j)$ respectively.

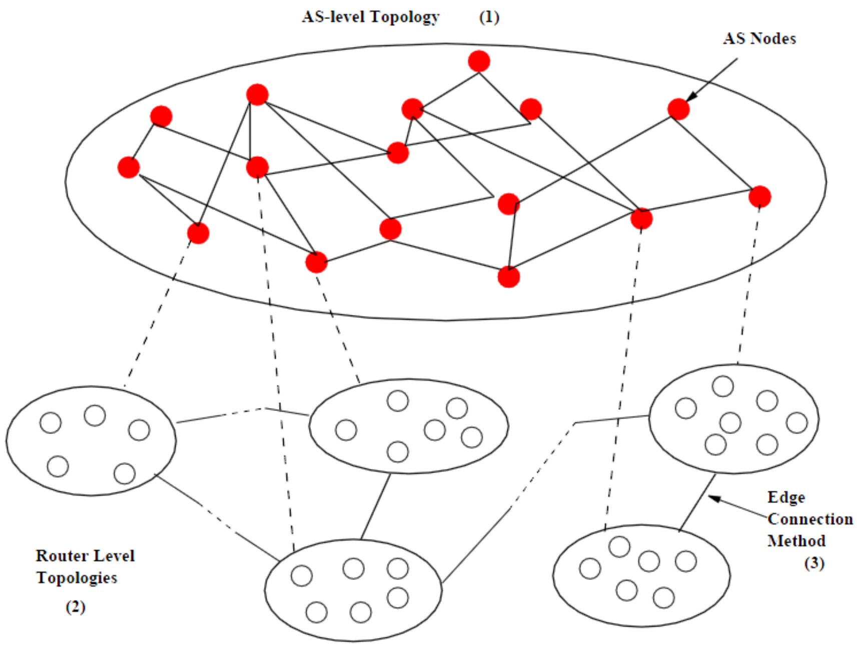


Figure 34: The structure of Top-down Hierarchical Models.

4.2.2 Simulation Results and Comparisons in GT-ITM Random Model

In this section, a statistical concept, called confidence interval, is employed in order to obtain reliable intervals of the simulation results for the Random and Semi-Random heuristics.

In statistics, a confidence interval (CI) is a particular kind of interval estimate of a population parameter. Instead of estimating the parameter by a single value, an interval likely to include the parameter is given. Thus, confidence intervals are used to indicate the reliability of an estimate. 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]

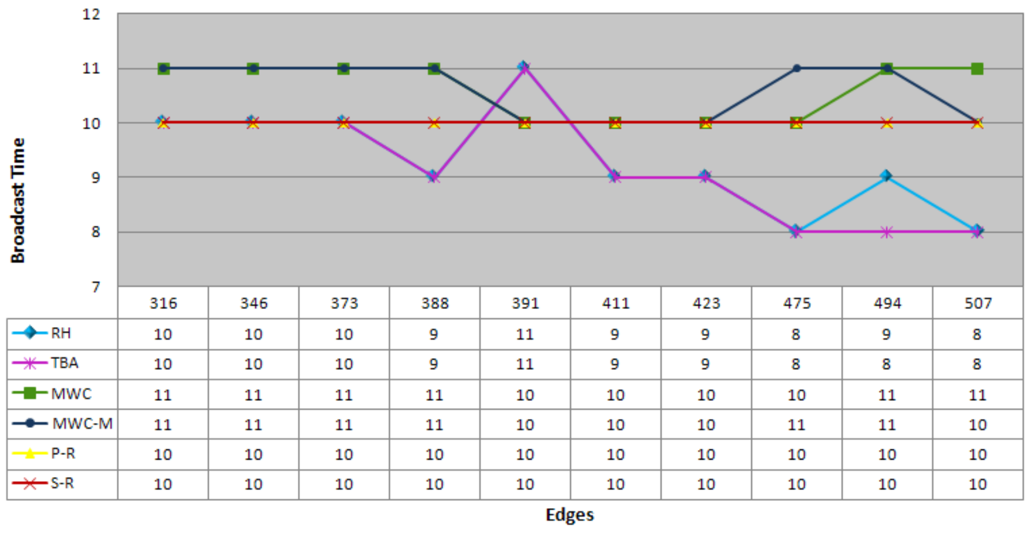
For each graph in the simulation, we perform both of the new heuristics 100 times. As a result, 100 samples for each heuristic are collected, then 99% confidence interval of the population is computed based on the sample mean and sample standard deviation. In the following tables, $99\%CI$ denotes the 99% confidence interval. As confidence interval is based on the sample mean, it is always shown in the form of an interval around the mean. If all samples are of the same value, there is no need to compute confidence interval, then just leave that entry blank.

At beginning, we simulate heuristics in GT-ITM Random model. Table 7 and Figure 35 show the data collected in the graph with 200 vertices. Here, P stands for the probability of having an edge between each pair of vertices. Obviously, higher probability leads to more edges in the graph. Six heuristics are abbreviated by RH, TBA, MWC, $MWC-M$, P-R, and S-R.

We can see that the Random and Semi-Random algorithms have identical simulation results, even though the number of edges rises from 316 to 507. However, referring to their confidence interval, the Semi-Random heuristic always works a bit better than the Random one. Compared with the Round_Heuristic and the Tree Based Algorithm, the new heuristics only generate better results in the graph of 391 edges. In Figure 35, we can see that the MWC and MWC-Modified heuristics have the worst results.

P	Edge	RH	TBA	MWC	MWC-M	P-R	99%CI	S-R	99%CI
0.015	316	10	10	11	11	10	10.5-10.75	10	10.1-10.31
0.016	346	10	10	11	11	10	10.78-10.97	10	10.33-10.58
0.017	373	10	10	11	11	10	10.52-10.77	10	10.26-10.51
0.018	388	9	9	11	11	10	10.66-10.89	10	10.33-10.58
0.019	391	11	11	10	10	10	10.6-10.83	10	10.4-10.65
0.02	411	9	9	10	10	10	10.28-10.53	10	10.09-10.3
0.022	423	9	9	10	10	10	10.41-10.66	10	9.99-10.1
0.024	475	8	8	10	11	10	10.87-11.06	10	10.86-10.99
0.025	494	9	8	11	11	10	10.88-11.11	10	10.48-10.73
0.026	507	8	8	11	10	10	10.94-11.07	10	10.96-11.01

Table 7: Simulation results in GT-ITM Random model with 200 vertices.



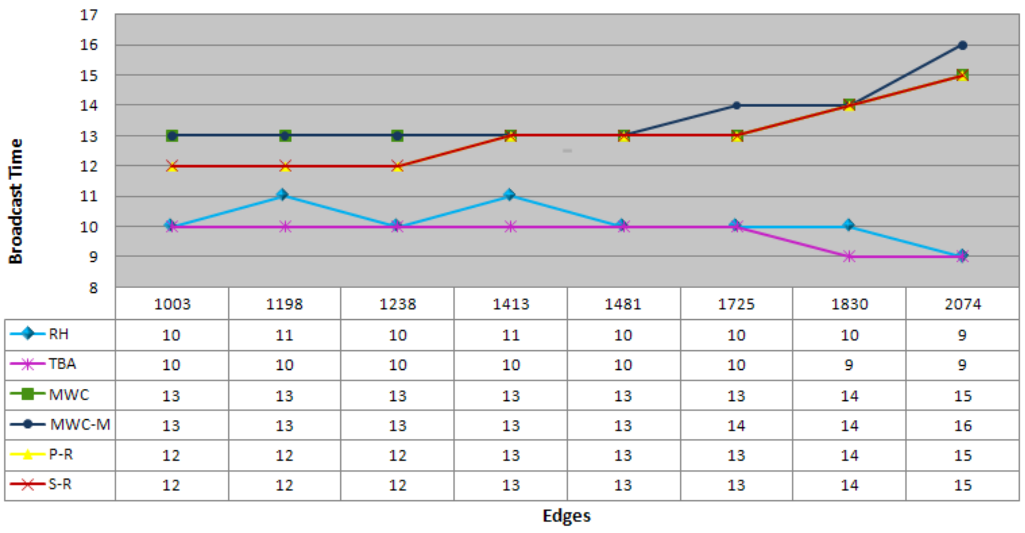
	316	346	373	388	391	411	423	475	494	507
RH	10	10	10	9	11	9	9	8	9	8
TBA	10	10	10	9	11	9	9	8	8	8
MWC	11	11	11	11	10	10	10	10	11	11
MWC-M	11	11	11	11	10	10	10	11	11	10
P-R	10	10	10	10	10	10	10	10	10	10
S-R	10	10	10	10	10	10	10	10	10	10

Figure 35: Simulation results in GT-ITM Random model with 200 vertices.

Table 8 and Figure 36 concern the GT-ITM Random model with 500 vertices. In this case, the Random and Semi-Random heuristics still have the same simulation results. However, with the number of edges increasing, their simulation results climb up slowly. Meanwhile, the results of the Round_Heuristic and Tree Based Algorithm fluctuate around 10. At last, when the number of edges is equal to 2074, the results of the new heuristics are almost twice that of the Round_Heuristic and the Tree Based Algorithm. Actually, the bigger probability p increases the number of edges, which leads to the increasement of degree of most vertices in the graphs. After performing breadth-first search, more vertices will be on the same layer, which increases the total broadcast time. Figure 36 shows clearly that the Round_Heuristic and the Tree Based Algorithm have better simulation results.

P	Edge	RH	TBA	MWC	MWC-M	P-R	99%CI	S-R	99%CI
0.008	1003	10	10	13	13	12	12.18-12.41	12	12.1-12.31
0.009	1198	11	10	13	13	12	12.78-12.97	12	12.76-12.97
0.01	1238	10	10	13	13	12	12.82-12.97	12	12.86-12.99
0.011	1413	11	10	13	13	13	13.02-13.19	13	12.96-13.03
0.012	1481	10	10	13	13	13	13.72-13.99	13	13.24-13.49
0.014	1725	10	10	13	14	13	13.81-13.98	13	13.76-13.95
0.015	1830	10	9	14	14	14	14.09-14.3	14	14.05-14.24
0.016	2074	9	9	15	16	15	15.08-15.31	15	15.31-15.56

Table 8: 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

Figure 36: Simulation results in GT-ITM Random model with 500 vertices.

4.2.3 Simulation Results and Comparisons in GT-ITM Transit-Stub Model

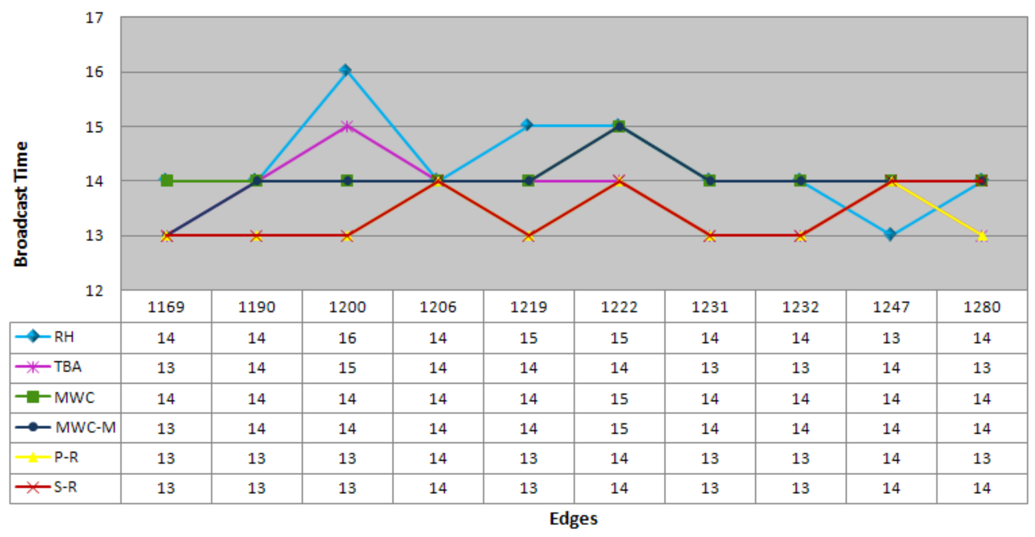
The first kind of GT-ITM Transit-Stub models that we studied are generated by following parameters. The initial seed is 47. Each graph has 3 stub domains per transit node, with no extra transit-stub or stub-stub edges. There are 3 transit domains, each of which has 8 nodes, and an edge between each pair of nodes with probability 0.5. Meanwhile, each stub domain has (on average) 8 nodes, and edge probability is also 0.5. Thus, the graphs have $3 \times 8 \times (1 + 3 \times 8) = 600$ vertices.

Table 9 and Figure 37 show the results in ten such graphs of different edges. The blank in the table means that all samples are of the same value, and there is no need to compute confidence interval. For example, when edge number is 1280, the results of Semi-Random algorithm are consistent, and do not have interval. With the number of edges increasing, the simulation results of all six heuristics fluctuate between 13 and 16. The Random and

Semi-Random heuristics almost have the same results except in the last case. However, if considering the confidence interval, the Semi-Random one works better than the Random one. Unlike the situations in GT-ITM Random model, the two new heuristics work best in this model, and the other heuristics are more or less on the same level, that can be observed in Figure 37.

Edge	RH	TBA	MWC	MWC-M	P-R	99%CI	S-R	99%CI
1169	14	13	14	13	13	13.34-13.59	13	13.2-13.45
1190	14	14	14	14	13	14.08-14.39	13	14.27-14.78
1200	16	15	14	14	13	13.89-14	13	13.13-13.68
1206	14	14	14	14	14	13.98-14.09	14	14-14.13
1219	15	14	14	14	13	13.46-13.71	13	13.21-13.46
1222	15	14	15	15	14	14.85-15.12	14	14.75-14.94
1231	14	13	14	14	13	13.78-13.97	13	13.51-13.76
1232	14	13	14	14	13	14-14.21	13	13.55-13.8
1247	13	14	14	14	14	-	14	-
1280	14	13	14	14	13	14-14.17	14	-

Table 9: 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

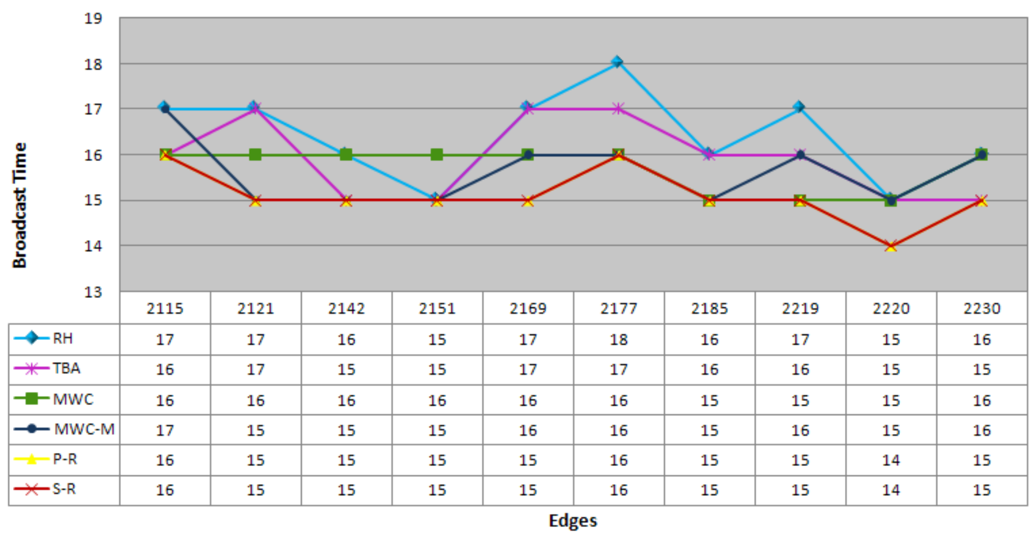
Figure 37: Simulation results in GT-ITM Transit-Stub model with 600 vertices.

Table 10 and Figure 38 show the simulation results in another kind of GT-ITM Transit-Stub model, starting with initial seed 47. Each graph has 4 stub domains per transit node, with no extra transit-stub or stub-stub edges. There are 4 transit domains, each of which has 8 nodes, and an edge between each pair of nodes with probability 0.5. Meanwhile, each stub domain has (on average) 8 nodes, and edge probability is also 0.5. Thus, the graphs have $4 \times 8 \times (1 + 4 \times 8) = 1056$ vertices.

This time, the Random and Semi-Random heuristic have the same simulation results, and both of them generate better results than the other heuristics. Among the other four heuristics, it is hard to tell which one performs the best or the worst. (See Figure 38).

Edge	RH	TBA	MWC	MWC-M	P-R	99%CI	S-R	99%CI
2115	17	16	16	17	16	16.23-16.48	16	16.25-16.5
2121	17	17	16	15	15	15.57-15.81	15	15.08-15.27
2142	16	15	16	15	15	15.84-16.13	15	15.42-15.67
2151	15	15	16	15	15	15.19-15.42	15	-
2169	17	17	16	16	15	15.86-16.01	15	15.6-15.83
2177	18	17	16	16	16	16.39-16.64	16	16.23-16.48
2185	16	16	15	15	15	-	15	-
2219	17	16	15	16	15	15.29-15.54	15	15.33-15.58
2220	15	15	15	15	14	15.03-15.24	14	15.03-15.26
2230	16	15	16	16	15	15.72-15.91	15	15.4-15.65

Table 10: 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

Figure 38: Simulation results in GT-ITM Transit-Stub model with 1056 vertices.

4.2.4 Simulation Results and Comparisons in Tiers Model

All the heuristics are simulated in two Tiers models, one of which has 355 vetices, while the other has 1105 vertices. The graphs of 355 vertices consist of one WAN, ten MANs and five LANs, while graphs of 1105 vertices are constituted by one WAN, ten MANs and ten LANs. All parameters used to generate these two kinds of graphs are listed in Table 11 and Table 12.

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 11: Parameters for Tiers model with 355 vertices.

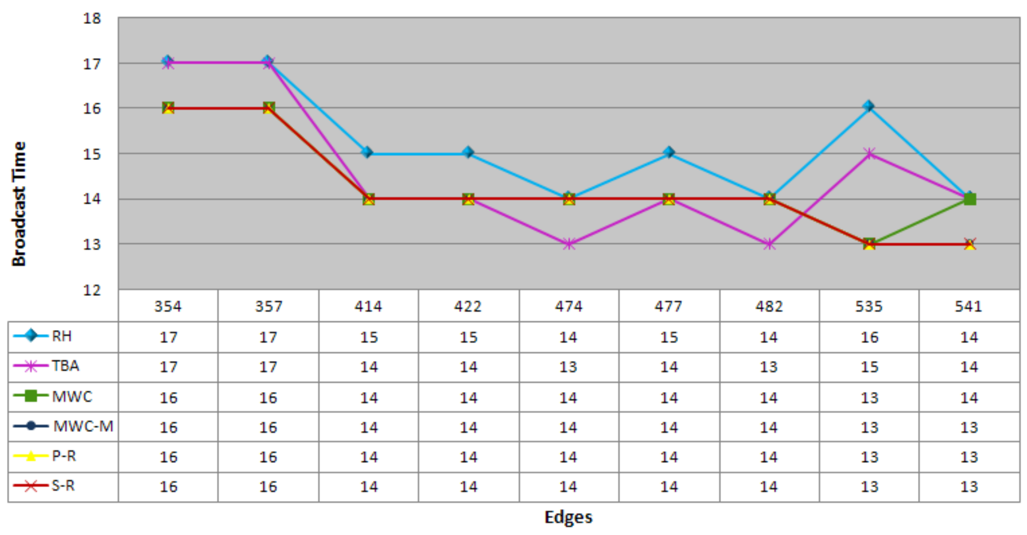
Edge	N_W	N_M	N_L	S_W	S_M	S_L	R_W	R_M	R_L	R_{MW}	R_{LM}
1214	1	10	10	5	10	10	1	1	1	2	2
1324	1	10	10	5	10	10	1	1	1	3	3
1447	1	10	10	5	10	10	1	1	1	4	4
1106	1	10	10	5	10	10	2	2	1	1	1
1216	1	10	10	5	10	10	2	2	1	2	2
1326	1	10	10	5	10	10	2	2	1	3	3
1110	1	10	10	5	10	10	3	2	1	1	1
1220	1	10	10	5	10	10	3	2	1	2	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 12: Parameters for Tiers model with 1105 vertices.

Table 13 and Figure 39 show the simulation results in Tiers graphs of 355 vertices. Although both Random and Semi-Random heuristics have the same minimal results, the Semi-Random one works better than the Random one when considering confidence intervals. In Figure 39, we can see clearly that all these heuristics fluctuate a lot, and it is hard to point out which one works the best. However, the new heuristics have the advantage of having lower time complexity.

Edge	RH	TBA	MWC	MWC-M	P-R	99%CI	S-R	99%CI
354	17	17	16	16	16	-	16	-
414	15	14	14	14	14	-	14	-
474	14	13	14	14	14	14.21-14.4	14	-
357	17	17	16	16	16	-	16	-
477	15	14	14	14	14	14.2-14.37	14	-
535	16	15	13	13	13	13.68-13.93	13	13.52-13.77
422	15	14	14	14	14	14.62-14.89	14	-
482	14	13	14	14	14	14.37-14.6	14	-
541	14	14	14	13	13	13.72-13.89	13	13.43-13.68

Table 13: The minimum simulation results in Tiers model with 355 vertices.



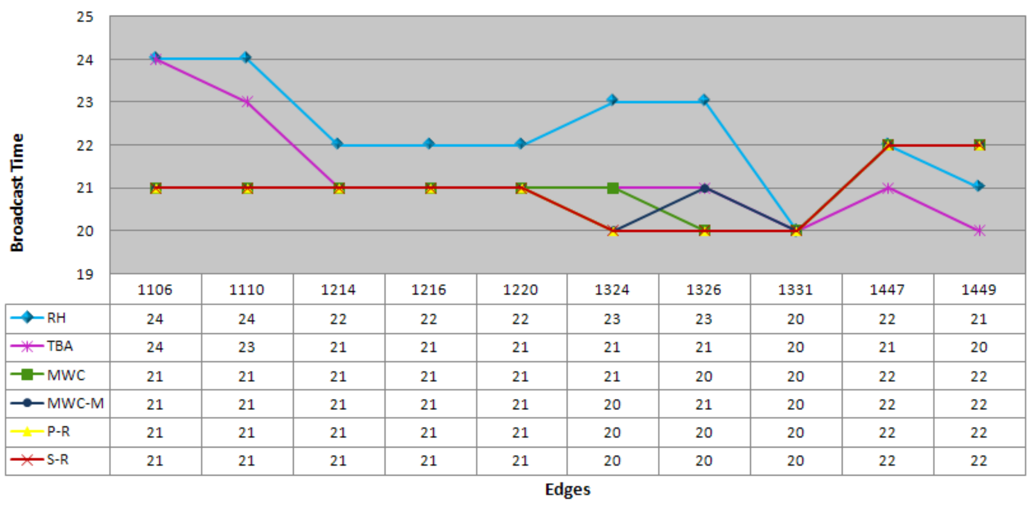
	354	357	414	422	474	477	482	535	541
RH	17	17	15	15	14	15	14	16	14
TBA	17	17	14	14	13	14	13	15	14
MWC	16	16	14	14	14	14	14	13	14
MWC-M	16	16	14	14	14	14	14	13	13
P-R	16	16	14	14	14	14	14	13	13
S-R	16	16	14	14	14	14	14	13	13

Figure 39: Simulation results in Tiers model with 355 vertices.

Table 14 and Figure 40 provide the simulation results in Tiers graphs with 1105 vertices. The Semi-Random heuristic has lower confidence intervals than the Random one, although they have identical results. The MWC and MWC-Modified heuristics also have similar simulation results. Comparing all the heuristics, the Round_Heuristic and Tree Based Algorithm only work better in the graph with 1449 edges. Figure 40 shows clearly that the new heuristics performs better except for the last two points.

Edge	RH	TBA	MWC	MWC-M	P-R	99%CI	S-R	99%CI
1214	22	21	21	21	21	-	21	-
1324	23	21	21	20	20	20.5-20.81	20	-
1447	22	21	22	22	22	-	22	-
1106	24	24	21	21	21	-	21	-
1216	22	21	21	21	21	-	21	-
1326	23	21	20	21	20	20.66-20.95	20	20.14-20.37
1110	24	23	21	21	21	-	21	-
1220	22	21	21	21	21	-	21	-
1331	20	20	20	20	20	20.13-20.38	20	-
1449	21	20	22	22	22	21.99-22.12	22	-

Table 14: The minimum simulation results in Tiers model with 1105 vertices.



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

Figure 40: Simulation results in Tiers model with 1105 vertices.

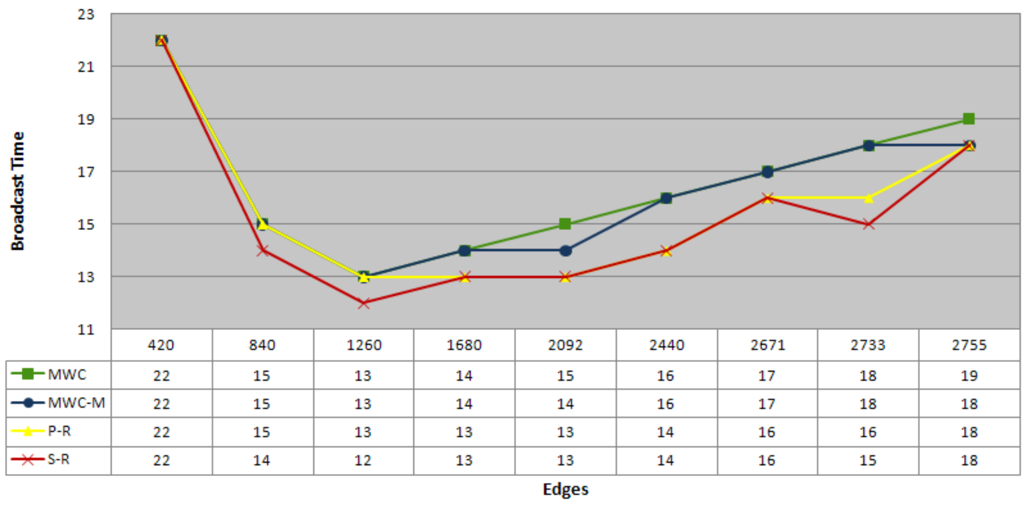
4.2.5 Simulation Results and Comparisons in BRITE Top-down Hierarchical Model

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

Table 15 and Table 16 show the simulation results in the graphs with 400 vertices. Table 15 is for the Waxman model, and Table 16 is for the Barabasi-Albert model. Results in these two models are similar. With the number of edges increasing, the results of the four heuristics decline first, and then ascend slowly. From Figures 41 and 42, we can see that the Semi-Random algorithm performs the best among all the four heuristics.

Edge	MWC	MWC-M	P-R	99%CI	S-R	99%CI
420	22	22	22	-	22	-
840	15	15	15	15.12-15.53	14	14.97-15.1
1260	13	13	13	13.64-13.93	12	12.94-13.15
1680	14	14	13	13.21-13.52	13	13.08-13.31
2092	15	14	13	13.52-13.81	13	13.79-13.96
2440	16	16	14	14.96-15.25	14	14.19-14.44
2671	17	17	16	16.38-16.65	16	16.35-16.6
2733	18	18	16	16.03-16.34	15	15.76-15.97
2755	19	18	18	19.17-19.52	18	-

Table 15: Simulation results in BRITE Top-down Waxman model with 400 vertices.

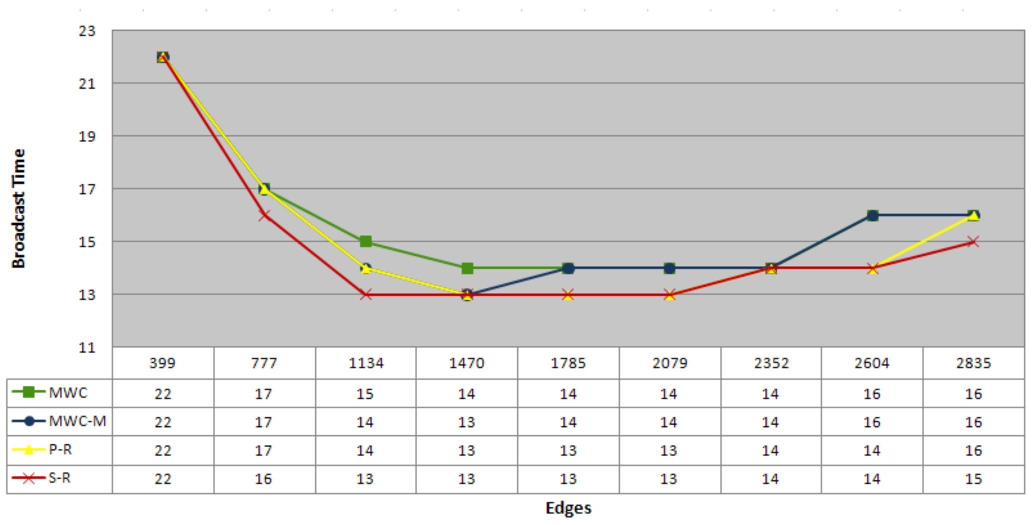


	420	840	1260	1680	2092	2440	2671	2733	2755
MWC	22	15	13	14	15	16	17	18	19
MWC-M	22	15	13	14	14	16	17	18	18
P-R	22	15	13	13	13	14	16	16	18
S-R	22	14	12	13	13	14	16	15	18

Figure 41: Simulation results in BRITE Top-down Waxman model with 400 vertices.

Edge	MWC	MWC-M	P-R	99%CI	S-R	99%CI
399	22	22	22	-	22	-
777	17	17	17	17-17.21	16	-
1134	15	14	14	15.05-15.4	13	13.95-14.08
1470	14	13	13	14.16-14.49	13	13.03-13.2
1785	14	14	13	13.8-14.13	13	-
2079	14	14	13	13.25-13.5	13	13.19-13.44
2352	14	14	14	14.09-14.3	14	14.26-14.6
2604	16	16	14	14.63-14.88	14	14.31-14.56
2835	16	16	16	16.15-16.46	15	14.98-15.09

Table 16: Simulation results in BRITE Top-down BA model with 400 vertices.



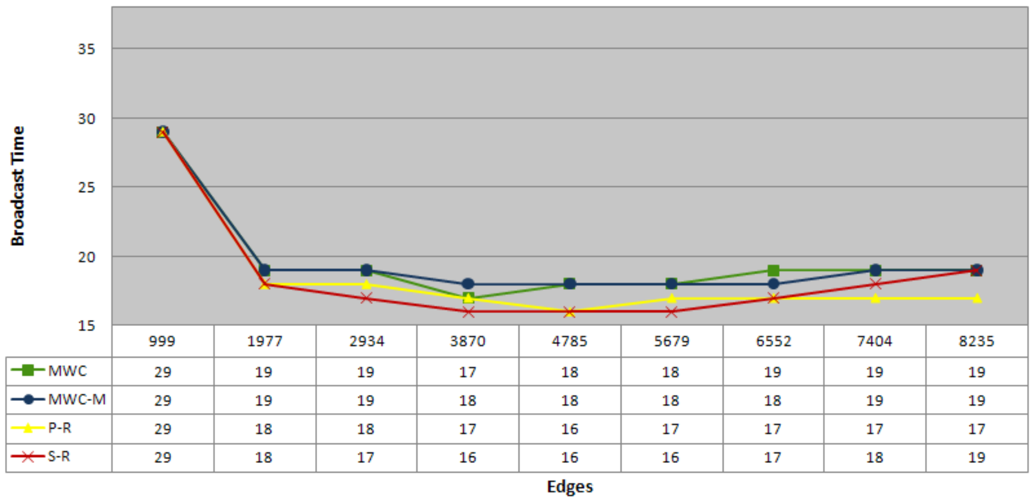
	399	777	1134	1470	1785	2079	2352	2604	2835
MWC	22	17	15	14	14	14	14	16	16
MWC-M	22	17	14	13	14	14	14	16	16
P-R	22	17	14	13	13	13	14	14	16
S-R	22	16	13	13	13	13	14	14	15

Figure 42: Simulation results in BRITE Top-down BA model with 400 vertices.

Table 17 shows the simulation results in the Top-down Waxman model with 1000 vertices. Except for the graphs with 1020 edges, all the four heuristics fluctuate around 17 and 18. Figure 43 shows clearly that the new heuristics have better results than the MWC and MWC-Modified heuristics.

Edge	MWC	MWC-M	P-R	99%CI	S-R	99%CI
1020	29	29	29	-	29	-
2040	19	19	18	18.92-19.09	18	18.74-18.93
3060	19	19	18	18.52-18.77	17	17.9-18.03
4080	17	18	17	17.47-17.76	16	16.6-16.85
5100	18	18	16	17-17.33	16	16.26-16.55
6108	18	18	17	17.89-18.36	16	17.06-17.29
7116	19	18	17	17.92-18.39	17	17.64-17.87
8117	19	19	17	18.14-18.43	18	18.56-18.81
9122	19	19	17	17.83-18.08	19	-

Table 17: Simulation results in BRITE Top-down Waxman model with 1000 vertices.



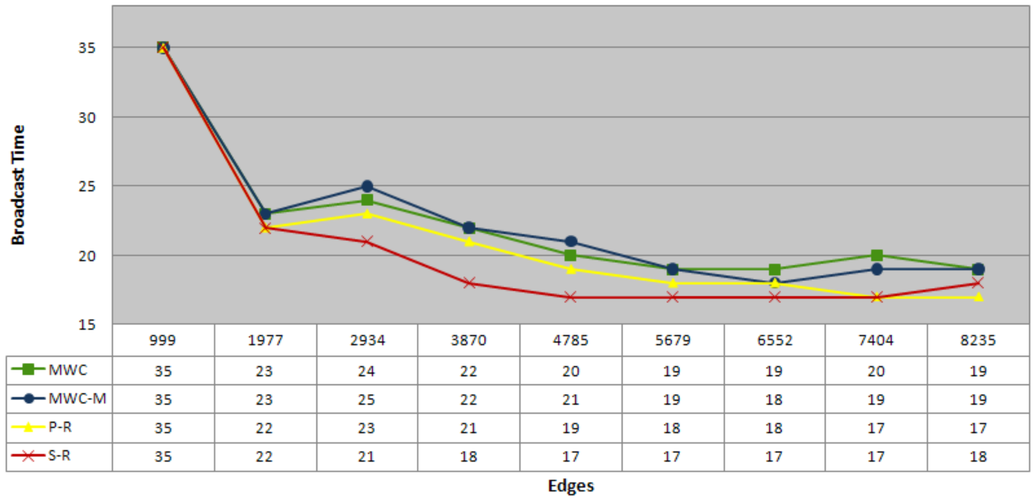
	999	1977	2934	3870	4785	5679	6552	7404	8235
MWC	29	19	19	17	18	18	19	19	19
MWC-M	29	19	19	18	18	18	18	19	19
P-R	29	18	18	17	16	17	17	17	17
S-R	29	18	17	16	16	16	17	18	19

Figure 43: Simulation results in BRITE Top-down Waxman model with 1000 vertices.

Table 17 and Figure 44 show the simulation results in the Top-down Barabasi-Albert model with 1000 vertices. The Random heuristic only generates better results than the Semi-Random one in the graph with 8235 edges. The MWC and MWC-Modified heuristics have similar results, and both perform worse than the new heuristics.

Edge	MWC	MWC-M	P-R	99%CI	S-R	99%CI
999	35	35	35	-	35	-
1977	23	23	22	22.9-23	22	22.92-23.01
2934	24	25	23	23.95-24.3	21	21.75-21.98
3870	22	22	21	21.65-22.18	18	19.02-19.27
4785	20	20	19	20.73-21.22	17	18.18-18.47
5679	19	19	18	19.05-19.34	17	17.56-17.83
6552	19	18	18	18.39-18.8	17	17.53-17.82
7404	20	19	17	18.36-18.85	17	18.23-18.52
8235	19	19	17	18.23-18.62	18	18.922-19.31

Table 18: 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	21	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

Figure 44: Simulation results in BRITE Top-down BA model with 1000 vertices.

4.3 Summary

Based on our extensive simulations in different topologies and network models, we can conclude that in most cases the Random heuristic almost has the same performance as the Semi-Random one. In commonly used topologies, they do not work as well as the Round_Heuristic(RH) and the Tree Based Algorithm(TBA), especially in the Hypercube graphs. That is because of the high degree of each vertex in the graph and the property that each vertex is informed via shortest path. However, in Cube Connected Cycles and Shuffle-Exchange graphs, the performance of our heuristics are comparable to RH and TBA. The situation is quite different in network models. In particular, the new heuristics generate better results in graphs of GT-ITM Transit-Stub model and Tiers model.

Moreover, the time complexities of the new heuristics are always $O(|E|)$, which is obviously much lower than that of the other heuristics. The Round_Heuristic is of complexity $O(R|V|^2|E|)$, while the Tree Based Algorithm has time complexity $O(R|E|)$, where R is the total broadcast time. Thus, in the worst case, their time complexities are $O(|V|^3|E|)$ and $O(|V||E|)$ respectively. The time complexity of MWC algorithm mostly depends on the different maximal flow algorithms, and from Table 1 we know that all their complexities are much higher than $O(|E|)$. The lower complexity makes the new heuristics be able to work for large graphs, and increase the possibility to obtain a better broadcast time in the same period of time.

Another advantage is that both of the new heuristics are simple to implement. There is no need to apply any complicated matching algorithm, which is the guarantee of their low complexities. As described in Chapter 2, the maximum weighted matching is a crucial procedure for the Round_Heuristic, and the MWC algorithm depends on the maximal flow algorithms. Both matching and maximal flow algorithms are not easy to implement, and lead to higher complexities.

Finally, the performance of the Round_Heuristic and Tree Based Algorithm heavily depends on the choice of some parameters. However, there is not any rule that guides how to determine the best parameters, and to obtain a better result one should simulate great amount of different values. This is not the case with our heuristics.

In conclusion, both of the new heuristics are easy to implement, have lower time complexities, and generate comparable results as the best existing heuristics in practice. Also, they generate the broadcast scheme where every vertex receives the message via a shortest path.

5 Conclusion and Future Work

Parallel and distributed computer systems are being used more and more. It is often the case that the communication latency is the bottleneck of these systems. Broadcasting is one of the elementary communication primitives, which implies that having an efficient broadcast protocol is necessary to having high performance parallel systems.

This thesis focuses on the problem to design algorithms that can generate efficient broadcast time for given graphs. This thesis focuses on this problem. Since the above problem has been proved to be NP-Complete for arbitrary graphs, we studied approximation and heuristic algorithms for the broadcast time problem in arbitrary graphs.

In this thesis, two new heuristics are presented for broadcasting in arbitrary networks. Based on the layer graph, the heuristics generate a spanning tree by random or semi-random matching strategies between each pair of adjacent layers. The final broadcast scheme comes from the spanning tree, through which every vertex in the network receives the message via shortest path. This property of the new heuristics can be applied in hop-limited systems, load balancing systems, or some systems demanding a small total number of message duplications.

By observing their performance in the commonly used topologies and network models, we see that the new heuristics are not suitable for graphs where most of the vertices have high degree. However, in the real networks few vertices have high degree, while most have low (even constant) degree. Based on our extensive simulations, we conclude that the new heuristics perform quite well in those models representing real networks. Especially, in GT-ITM Transit-Stub and Tiers models, the new heuristics provide better broadcast time. We can conclude that the new heuristics perform well in practice.

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

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

References

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

[2] A. Bar-Noy, S. Guha, J. Naor and B. Schieber. Multicasting in heterogeneous networks, *Proceedings of ACM symposium on Theory of Computing, STOC'98*, 1998.

[3] D. Barth and P. Fraigniaud. Approximation algorithms for structured communication problems, *ACM Symposium on Parallel Algorithms and Architectures, SPAA'97*, pp. 180-188, 1997.

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

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

[6] J.-C. Bermond and C. Peyrat. Broadcasting in de Bruijn networks, *Proceedings of the 19th S-E conference on Combinatorics, Graph Theory and Computing, Congressus Numerantium*, 66:283-292, 1988.

[7] G. T. Chen. An algorithm for gossiping and broadcasting, Master Thesis, Computer Science Department, Concordia University, 2006.

[8] M. J. Dinneen. The complexity of broadcasting in bounded-degree networks, *Computer Research and Appications*, Los Alamos National Laboratory, New Maxico 87545, U.S.A., 1994.

[9] M. B. Doar. A better model for generating test networks, *Proceedings of Global Telecommunications Conference, GLOBECOM'96*, pp. 86-93, 1996.

[10] M. Elkin and G. Kortsarz, A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem, *Proceedings of ACM symposium on Theory of Computing, STOC'02*, pp. 438-447, 2002.

[11] M. Elkin and G. Kortsarz. Sublogarithmic approximation for telephone multicast: path out of jungle, *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, Baltimore, MA*, pp. 76-85, 2003.

[12] A. M. Farley and S. T. Hedetniemi. Broadcasting in grid graphs, *Proceedings of Ninth S-E Conference on Combinatorics, Graph Theory and Computing. Utilitas Mathematica,Winnipeg*, pp. 275-288, 1978.

[13] U. Feige, D. Peleg, P. Raghavan and E. Upfal. Randomize broadcast in networks, *Proceedings of International Symposium on Algorithms, SIGAL'90*, 450:128-137, 1990.

[14] P. Fraigniaud and E. Lazard. Methods and problems of communication in usual networks, *Discrete Applied Mathematics*, 53:79-133, 1994.

[15] P. Fraigniaud and S. Vial. Approximation algorithms for broadcasting and gossiping, *Parallel Distributed Comput*, 43(1):47-55, 1997.

[16] P. Fraigniaud and S. Vial. Heuristic algorithms for personalized communication problems in point-to-point networks, *Proceedings of the 4th Colloquium on Structural Information and Communication Complexity, SIROCCO'97*, pp. 240-252, 1997.

[17] P. Fraigniaud and S. Vial. Comparison of heuristics for one-to-all and all-to-all communication in partial meshes, *Parallel Processing Letters*, 9(1):9-20, 1999.

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

[19] F. Guo, Finding the minimum broadcast time of bounded degree networks by backtracking, Master thesis, University of Auckland, 2001.

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

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

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

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

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

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

[26] A. Jakoby, R. Reischuk and C. Schindelhauer. The complexity of broadcasting in planar and decomposable graphs, *Discrete Applied Mathematics*, 83:179-206, 1998.

[27] Klaus Jansen and Haiko Muller. The minimum broadcast time problem for several processor networks, *Theoretical Computer Science*, 147(1,2):69-85, 1995.

[28] R. Klasing, B. Monien, R. Peine and E. A. Stohr. Broadcasting in butterfly and de-Bruijn networks. *Discrete Applied Mathematics*, 53:183-197, 1994.

[29] G. Kortsarz and D. Peleg. Approximation algorithms for minimum time broadcast, *SIAM Journal on Discrete Mathematics*, pp. 401-427, 1995.

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

[31] A. Liestman and J. Peters. Broadcast networks of bounded degree, *SIAM Journal on Discrete Mathematics*, 1:531-540, 1988.

[32] A. Medina, A. Lakhina, I. Matta and J. Byers. BRITE:Universal Topology Generation from a User's Perspective, *Technical Report BUCS-TR-2001-003*, 2001.

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

[34] C. D. Morosan. Studies of interconnection networks with applications in broadcasting, PHD. Thesis, Computer Science Department, Concordia University, 27-35, 2007.

[35] R. Ravi. Rapid rumor ramification: approximating the minimum broadcast time, *Proceedings of 35th Annual Symposium on Foundation of Computer Science*, pp. 202-213, 1994.

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

[37] P. Scheuermann and G. Wu. Heuristic algorithms for broadcasting in point-topoint computer network, *IEEE Transactions on Computers*, c-33(9):804-811, 1984.

[38] C. Schindelhauer. Broadcasting time cannot be approximated within a factor of $57/56 - \epsilon$, *ICSI Technical Report TR-00-002*, 2002.

[39] B. Shao. A heuristic for broadcasting in arbitrary networks, Master Thesis, Computer Science Department, Concordia University, pp. 55-57, 2003.

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

[41] Wikipedia: the free encyclopedia. Confidence Interval, available at *http://en.wikipedia.org/wiki/Confidence_interval*, last visited in July, 2010.

[42] Wikipedia: the free encyclopedia. NP(complexity), available at *http://en.wikipedia.org/wiki/NP_(complexity)*, last visited in July, 2010.

[43] Wikipedia: the free encyclopedia, NP-Complete, available at *http://en.wikipedia.org/wiki/NP-complete*, last visited in July, 2010.

[44] E. W. Zegura, K. Calvert and S. Bhattacharjee. How to model an internetwork, *Proceedings of Fifteenth Annual Joint Conference of the IEEE Computer Societies, INFOCOM'96*, 2:594-602, 1996.