LOW-COST DELAY-CONSTRAINED MULTICAST ROUTING HEURISTICS AND THEIR EVALUATION

By

VENKATA SRINIVAS IRAVA

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

AUGUST 2006

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of VENKATA SRINIVAS IRAVA find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

ACKNOWLEDGEMENT

I am eternally thankful to my advisor Dr. Carl Hauser for his help throughout my PhD program. His constant support and guidance have helped me immensely in my studies here at WSU. I would also like to thank my committee members Dr. David Bakken and Dr. Muralidhar Medidi for their valuable advice and support.

I would also like to thank to my parents Anuradha and Ravindra Babu and my sister Tulasi for their constant encouragement and for their trust me. Without their love and support this dissertation would not have been feasible. I am also thankful to all my friends here in Pullman.

I would also like to acknowledge and thank the different organizations that have funded my PhD studies at Washington State University. This dissertation was supported in part by NSF grants CCR-0326006, and CNS-0524695, and by assistanships from the school of Electrical Engineering and Computer Science, Washington State University.

ATTRIBUTION

Portions of this dissertation appear in the following publications.

- V.S. Irava, and C. Hauser, "Survivable low-cost low-delay multicast trees," Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), Nov. 2005

- C.H. Hauser, D.E. Bakken, I. Dionysiou, K.H. Gjermundrod, V.S. Irava, J. Helkey, and A. Bose, "Security, trust and QoS in next-generation control and communication for large power systems," International Journal of Critical Infrastructures, Inderscience, to appear, 2007.

- C.H. Hauser, D.E. Bakken, I. Dionysiou, K.H. Gjermundrod, V.S. Irava, and A. Bose, "Security, trust and QoS in next-generation control and communication for large power systems," International Workshop on Complex Network and Infrastructure Protection, Rome, March 28-29, 2006.

- V.S. Irava, and C.H. Hauser, "Low-cost delay-constrained survivable multicast routing heuristics," International Journal of Communication Systems, (submitted)

LOW-COST DELAY-CONSTRAINED MULTICAST ROUTING HEURISTICS AND THEIR EVALUATION

Abstract

by Venkata Srinivas Irava, Ph.D.
Washington State University
August 2006

Chair: Carl H. Hauser

Critical wide-area infrastructures (CWI) need real-time monitoring and control for their reliable operation and for fast identification and resolution of anomalies within these infrastructures. Real-time monitoring and control is made possible by the use of multicast routing for fast, efficient, and reliable dissemination of status information in the CWI.

The fast and efficient dissemination of status information in these infrastructures imposes delay and cost requirements on the multicast routing heuristic. To ensure reliability in the face of network failures the multicast graphs constructed by the heuristic need to have at least two paths from the source to each destination in the network.

Satisfying the delay requirements of the application ensures that the data delivered to the destination is fresh. Reducing the total cost of the multicast graph helps in lowering resource usage. Additionally, having two node-disjoint paths from the source to each destination makes a multicast graph resilient to node or edge failures and improves its survivability against such failures.

This dissertation presents new heuristics for constructing survivable and non-survivable low-cost delay-constrained multicast graphs and presents more extensive evaluations than have typically been used to demonstrate properties of multicast routing heuristics.

The edge-priority based dynamic weight heuristic (DWH) proposed in this work constructs

non-survivable, low-cost, delay-constrained multicast trees. DWH assigns dynamic weights to edges based on how close the edge is to violating the delay bound of the application. The effective-edge-costs (edge-costs influenced by edge-weights) are then used as indicators of the relative merit of including the edge in the paths from the source to the destination node.

Dynamic weight disjoint path pairs (DW-DPP) heuristic is the survivable variant of DWH and constructs delay influenced (like in DWH edges are assigned varying weights based on the delay bound requirements) shared disjoint path pairs from the source to each destination node. Edges can be shared by paths to multiple destination nodes thus reducing the total cost of the multicast graph.

The evaluations show that DWH and DW-DPP construct multicast graphs with $10-15\%$ lower costs than the node-priority based heuristics that are also explored in this work.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

Dedication

This dissertation is dedicated to my parents Anuradha and Ravindra Babu

and to my sister Tulasi

CHAPTER ONE

INTRODUCTION

Efficient and reliable operation of critical wide-area infrastructure (CWI) systems is realized through real-time monitoring and control of such systems. It is essential for the multiple CWI controllers to have a clear picture of its status at all times. A real-time picture of the CWI helps the controllers in better allocating network resources or to effectively and efficiently respond to the demands of the network. Such real-time monitoring and control requirements of these infrastructures are realized through the use of a multicast routing heuristic which can efficiently disseminate data through the network.

Multicast routing chooses paths in a network from a source to multiple destination nodes. Multicast routing heuristics construct multicast trees spanning the source node and the multicast destination nodes. Additionally, constrained multicast QoS routing heuristics construct multicast trees that satisfy path QoS requirements (such as meeting end-to-end delay bounds on paths from the source to destination nodes) and tree QoS requirements (such as constructing trees with minimal cost). Multicast graphs with disjoint paths to destinations are protected against failures along a single path and hence are said to be more survivable than ones that have a single path to each destination (non-survivable multicast trees). Node-survivable multicast graphs have node-disjoint paths to each destination while edge-survivable multicast graphs have edge-disjoint paths.

The problem of constructing a minimum-cost multicast tree, a Steiner tree, is NP-complete [1]. Several Steiner tree heuristic solutions have been proposed in the past [2], [3], and [4]. Since the simpler Steiner tree problem is NP-complete the problem of constructing a minimum-cost, delay-constrained multicast graph (either non-survivable or survivable) is also NP-complete and hence can only be solved by heuristic solutions (and not by algorithmic ones).

1.1 Motivation

The electric power-grid is an example of a CWI system which requires real-time monitoring and control so that the supply and demand requirements of the grid are balanced at all times. Real-time monitoring and control of the grid is also useful in responding effectively and efficiently to anomalies in the grid, such as the loss of a power generation station.

GridStat, [5], [6], [7], and [8] is a publish-subscribe middleware framework being developed at Washington State University for efficient, secure and reliable dissemination of status information in the electric power grid. The dissemination of the status information from the publishers of such information in a reliable and timely manner to the subscribers enables the subscribers to have a real-time picture of the status of the grid. The dissemination of status information in these networks imposes delay, cost and path-redundancy requirements on the multicast routing heuristic.

Each subscriber in the GridStat framework is a computer application associated with a trusted utility organization. GridStat makes each status item available to all subscribers that request it. These subscribers are thus better able to respond to the demand and supply fluctuations within the grid ensuring robust and efficient operation of the grid. GridStat middleware framework is shown in Figure 1.1. The GridStat middleware as shown in the figure is situated above the operating system and provides a common programming framework for distributed applications situated above it.

One of the goals of GridStat is to provide QoS guarantees in data delivery through the use of an optimized multicast routing heuristic. The three basic requirements of any such multicast routing heuristic are:

1. The paths from the source to each destination must meet the delay bound of the application.

2. The multicast tree constructed by the multicast routing heuristic should have low tree cost.

3. The multicast tree should be constructed sequentially. That is the *online* heuristic should incrementally construct and expand the multicast tree as new destinations are added over

-

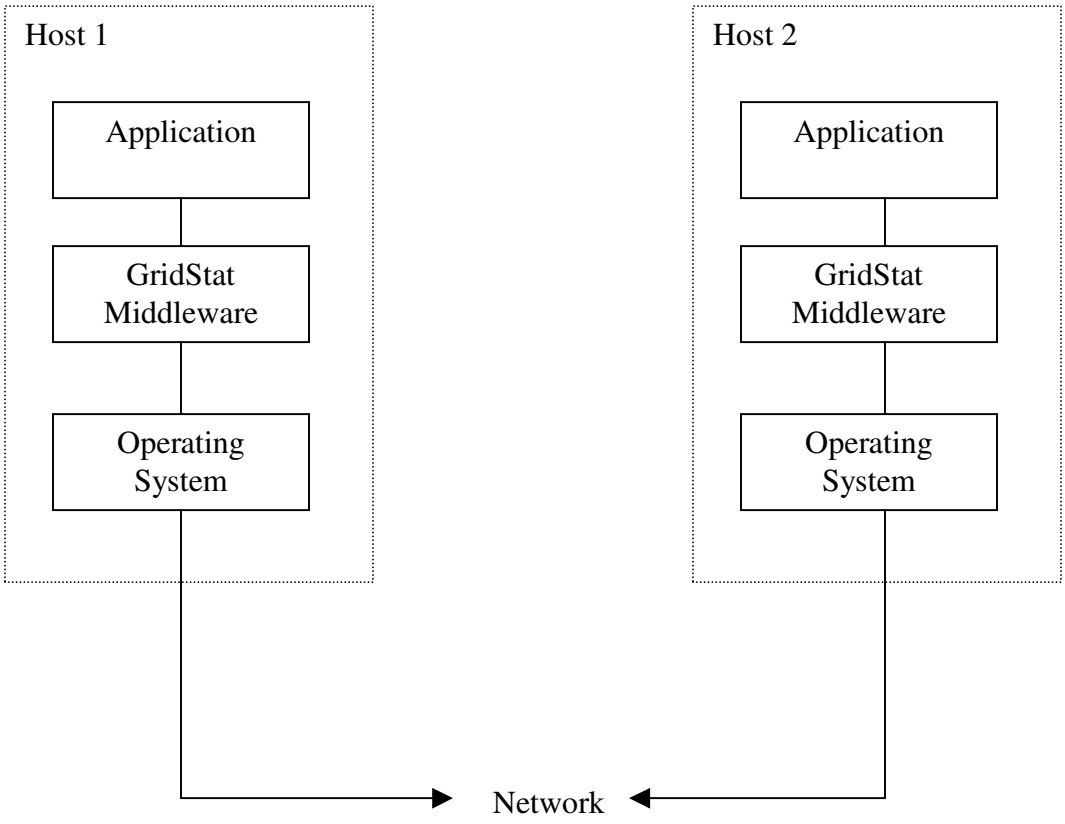

Figure 1.1: GridStat Middleware Framework

time. This is opposed to the *offline* scenario in which it is necessary for the publisher to identify all the subscribers of the multicast session ahead of time and before the start of the multicast session.

These three requirements will ensure the following three things for applications that use the GridStat framework:

1. Data are delivered in a timely manner to each subscriber.

2. The routing heuristic makes efficient use of available resources in the communication network.

3. Online multicast tree construction enables the subscribers to join the multicast tree at the point where they need the status data. Therefore it is not necessary for subscribers to synchronize their joining the multicast session with the other subscribers.

Low-Cost Delay-Constrained Multicast Routing Heuristics

Multicast routing heuristics in the literature do not address all the three requirements listed above. This dissertation proposes and extensively evaluates online, low-cost, delay-constrained multicast routing heuristics that satisfy these requirements.

This dissertation presents and evaluates new edge-priority-based heuristics: Dynamic Weight Heuristic (DWH) and its two variants Unit Weight Heuristic (UWH) and Zero Weight Heuristic (ZWH). In addition to these heuristics, new online variants of the node priority based QoS Dependent Multicast Routing (QDMR) heuristic proposed in [9] are also considered in the evaluations. All of these heuristics incrementally construct multicast trees that meet the delay requirements of the application while attempting to keep the tree cost low.

An additional goal of the GridStat framework is to have redundancy in the data delivery process. Redundancy in data delivery helps ensure that the data is delivered in a reliable manner to the subscribers. Having disjoint paths to each subscriber node makes multicast delivery resilient

against failures along a single path, enhancing the reliability of the framework. Therefore the multicast graph constructed needs to have disjoint paths to each destination.

Low-cost delay-constrained disjoint path pair heuristics for multicast routing

Disjoint path pair (DPP) multicast routing heuristics incrementally construct shared disjoint path pair multicast graphs. Applications delay requirements must be met on each of the redundant paths and as before low-cost is desirable. Note that in the case of disjoint path pairs the multicast graph is no longer a tree.

This dissertation presents and evaluates a new disjoint-path-pair variant of DWH called Dynamic Weight Disjoint Path Pairs (DW-DPP) and two variants: Unit Weight Disjoint Path Pairs (UW-DPP), and Zero Weight Disjoint Path Pairs (ZW-DPP). In addition to these heuristics, new disjoint-path-pair online variants of QDMR are also evaluated. All of these heuristics incrementally construct shared disjoint path pair graphs. Like before, the heuristics construct DPP graphs that meet the delay requirements of the application along both the redundant paths while trying to keep the DPP graph cost low.

Evaluation of low-cost delay-constrained multicast routing heuristics

Additionally, this dissertation is also motivated in part by the need for an effective evaluation technique for such multicast routing heuristics. Researchers in the past, [10], [11], [12], have resorted to evaluating such heuristics under a limited context, often focussing on small example graphs (< 20 nodes) that serve to illustrate the operation of the heuristic but which give little sense of its performance on bigger networks. These limited evaluations make it difficult to predict the performance of a heuristic relative to others and also its performance in a different environment.

Performance of the heuristics discussed here is determined by:

- Evaluating the heuristics' performance on several different graphs of the same type: comparing the heuristics' performance on several graphs of the same type but with varying node

degrees helps in identifying whether the heuristics' performance is affected by the node degree of the graph used in the evaluation.

- Evaluating the heuristics' performance on disparate graphs: a evaluation technique which satisfies this requirement helps predict the performance of a heuristic in a different environment. For example, it is difficult to predict the performance of a heuristic on a pure random graph[1] if the evaluation was only conducted on Waxman graphs [2].

- Evaluating the heuristics' performance on a large number of graphs: comparing the heuristics' performance over a large number of graphs of the same type helps in establishing the relative performance of the heuristics over a range of graphs.

- Comparing the heuristics' performance to the optimal solution. Experimental comparison to the optimal solution may not be feasible as the problem of constructing an optimized multicast tree is NP-complete, [1].

- Evaluating the heuristics' performance over a range of delay bound values: this requirement enables the heuristics' performance comparison over the range of the delay requirements of the application. Such evaluations reveal the cost performance of the heuristics as the delay requirements vary from easy to satisfy to hard to satisfy.

In this dissertation both the multicast tree and shared disjoint path pair low-cost delay-constrained multicast routing solutions are extensively evaluated based on the above listed evaluation techniques.

1.2 Problem Statement

The low-cost delay-constrained multicast routing problem is formulated in standard terms:

[1] A simple random graph generation method, [13], where edges are added probabilistically between nodes placed on a plane

[2] Waxman is a random graph generation method, [14], in which the probability of an edge existing between two nodes depends on the distance between the two nodes and on the user defined parameters α and β

- A network is represented as a graph $G = (N, E)$ where N is the node-set in the network and E is the edge-set in the network.
- The numbers of nodes and edges in the network are $|N|$ and $|E|$ respectively.
- The multicast destination set is M, and $m = |M|$ is the number of the multicast destination nodes in the network.
- The source and destination nodes in the graph are represented by s and d respectively.
- An edge in the network is represented by $e \in E$ or more explicitly an edge from node p to node q is represented by $e_{pq} \in E$.
- The cost and delay associated with an edge e_{pq} are $c_{e_{pq}}$ and $d_{e_{pq}}$ respectively.
- A node in the network is represented by $n \in N$ or more explicitly a node with a label p is represented by $n_p \in N$.
- The cost and delay associated with a node n_p are c_{n_p} and d_{n_p} respectively.
- The node-set N^D represents the nodes that need to be explored in the Dijkstra algorithm.
- A heuristic H denotes any low-cost, delay-constrained multicast routing heuristic.
- The multicast solution constructed by a multicast routing heuristic H is $G_{H_M} = (N_M, E_M)$ where $M \subseteq N_M$.
- The delay and cost from the source to a node d along the path P_{sd} are labelled as $D_{P_{sd}}$ and $C_{P_{sd}}$ respectively.
- The tree (or graph) cost of the multicast solution constructed by H, T_H, is the sum of the cost of edges in G_{H_M}, $\sum_{e \in E_M} c_e$.

- A path from node s to node d is an alternating sequence of nodes and edges and is labelled P_{sd}.

- A path pair (P^1_{sd}, P^2_{sd}) is node-disjoint $(P^1_{sd}, P^2_{sd})^{ND}$ if the paths have no common nodes (and no common edges). A path pair is edge-disjoint $(P^1_{sd}, P^2_{sd})^{ED}$ if the paths have no common edges (a less stringent requirement).

- The end-to-end delay of path P_{sd} is the sum of the delays on the edges in the path.

$$D_{P_{sd}} = \sum_{e \in P_{sd}} d_e$$

- Delay bound (Δ) is the maximum allowable end-to-end delay from the source to any multicast destination node in the tree.

- Satisfies delay bound (SatΔ) is a property of (G, H, M, Δ). If s is the source node:

$$\text{Sat}\Delta = \begin{cases} true & \text{if } D_{P_{sd}} \leq \Delta \quad (\forall\, d \in M); \\ false & \text{otherwise.} \end{cases}$$

- Sat$\Delta(d)$ is a property of (G, H, d, Δ). If s is the source node:

$$\text{Sat}\Delta(P_{sd}) = \begin{cases} true & \text{if } D_{P_{sd}} \leq \Delta; \\ false & \text{otherwise.} \end{cases}$$

The minimum-cost delay-constrained single path multicast routing problem: Given a network $G = (N, E)$ and a multicast destination set M find a multicast solution $G_M = (N_M, E_M)$ having a path P_{sd} for each $d \in M$ such that SatΔ(d) is 1 and $\sum_{e \in E_M} c_e$ is minimum.

The minimum-cost delay-constrained disjoint path multicast routing problem: Given a network $G = (N, E)$ and a multicast destination set M find a multicast solution $G_M = (N_M, E_M)$ having path pairs $(P^1_{sd}, P^2_{sd})^{ND}$ for each $d \in M$ where SatΔ is 1 for each path in every path pair, such that $\sum_{e \in E_M} c_e$ is minimum.

1.3 Contributions

1.3.1 Non-survivable online low-cost delay-constrained multicast routing heuristics

A non-survivable online low-cost, delay-constrained multicast routing heuristic called Dynamic Weight Heuristic (DWH) is presented and evaluated. DWH dynamically assigns a delay influenced weight coefficient $W_{e_{pq}}$ to each edge e_{pq} in the network. The shortest path algorithm in the first phase is run using the effective cost (cost influenced by the weight coefficient), $c_{e_{pq}} * W_{e_{pq}}$, on each edge and computes the lowest-effective-cost path. If the lowest-effective-cost path to a destination fails to meet the delay bound of the application then a second phase constructs and merges the least-delay path to that destination to the multicast tree (if the least-delay path satisfies the delay bound of the application).

Two extreme variants of DWH: UWH the least-cost-path variant and ZWH the low-total-cost variant are also used in the evaluations. The evaluations show that DWH on an average constructs multicast trees with about 10% lower cost than the node-priority based heuristics derived from QDMR that are also explored in this work.

1.3.2 Survivable online low-cost delay-constrained multicast routing heuristics

A survivable low-cost delay-constrained variant of DWH called Dynamic Weight Disjoint Path Pairs (DW-DPP) is presented and evaluated. DW-DPP incrementally constructs shared disjoint path pairs to each multicast destination node. Like before, DW-DPP assigns a weight coefficient to each edge in the network and constructs delay-influenced shared disjoint path pairs. If any path in the delay-influenced disjoint path pair to a destination fails to meet the delay bound of the application then the least-delay disjoint path pair to that destination is added to the multicast graph.

UW-DPP, the least-cost disjoint path pair variant of DW-DPP, and ZW-DPP the low-total-cost disjoint path pair variant of DW-DPP are also considered in the evaluations. The evaluations show that DW-DPP on an average constructs multicast graphs with about 15% lower cost than the node-priority based survivable heuristics derived from QDMR that are also explored in this work.

CHAPTER TWO

LITERATURE SURVEY

This chapter presents a literature survey of low-cost, delay-constrained multicast routing heuristics. Additionally, it also contains a survey of related work in the evaluation of multicast routing heuristics: on the selection of appropriate graph models for evaluation of routing heuristics, and on the structure and size of multicast destination sets for multicast routing heuristics. A description of the different metrics available for evaluating the performance of a multicast routing heuristic is also presented. The chapter also presents a technique for comparing a multicast routing heuristic's performance to the best achievable solution.

2.1 Comparison with other heuristics

An important aspect in the evaluation of any multicast routing heuristic is the comparison of the heuristic's performance to other heuristics which exist in the literature for the same problem space.

2.1.1 Non-survivable low-cost delay-constrained multicast routing heuristics

Several heuristics for problems closely related to the construction of non-survivable, low-cost, delay-constrained, multicast trees have been proposed in the past. In [15] an offline heuristic called Delay Variation Multicast Algorithm (DVMA) for the construction of multicast trees satisfying end-to-end delay and delay-variation (among the different destinations) constraints is presented. Here, least-delay paths to all destinations are first computed and the longest among them is selected as the initial multicast tree (to minimize the delay variation among the nodes). Other nodes are added to the multicast tree by finding the least-delay paths to them from the multicast tree. Tree cost is not considered as a metric in this heuristic.

A non-survivable offline delay-insensitive heuristic Destination Driven Multicast (DDMC) with the aim of reducing the total tree cost is presented in [16]. This heuristic computes paths on the premise that paths going through the destination nodes should have a higher priority than

other paths. DDMC also assumes knowledge of all the destination nodes prior to the construction of the multicast tree. The destination nodes are made to appear as new sources in paths to other destination nodes by re-setting their cost estimate from the source to zero during the execution of a shortest-path heuristic. This tends to reduce the over-all cost of the multicast tree. DDMC requires only one pass of a shortest path heuristic. After this pass the resulting spanning tree is trimmed so as to only include destination nodes as its leaves. However DDMC ignores delay.

The QDMR heuristic mentioned in Section 1.1 is an improved heuristic based on DDMC. It considers both delay and tree cost in computing multicast trees. In QDMR destination nodes are assigned higher but variable priority (unlike in DDMC) during the multicast tree construction. This variable priority in QDMR is based on how far the candidate destination node is from violating the delay bound. The farther the destination node is from violating the delay bound the higher its priority and the more likely it is to be included in paths to other destination nodes.

This delay-influenced sharing of paths helps in reducing the multicast tree cost. QDMR executes one pass of a shortest path heuristic. If however the priority-influenced least-cost paths to some destination nodes are unable to meet the delay bound of the application then the least-delay paths to these nodes (with SatΔ = 1) are merged to the multicast tree.

The extended Prim-Dijkstra tradeoff algorithm (EPDT), [11], is a variant of QDMR and assigns a bounded coefficient (the coefficient assigned is always ≤ 0.5) to the destination nodes in the tree. EPDT is based on the premise that the variable priority being assigned in QDMR is effective in including destination nodes such as d in paths to other destination nodes when $D_{P_{sd}}$ is far lower than Δ. EPDT assigns priorities to destination nodes based on the delays of their adjacent nodes in the shortest path heuristic, this ensures that destination nodes are assigned higher priority regardless of the delay bounds of the application. However this also results in better delay performance of QDMR (relative to EPDT) at lower delay bound values.

DDMC, QDMR, and EPDT assume knowledge of all the destination nodes before constructing the multicast tree. In this research work we assume that the destination nodes arrive in a sequential

fashion and hence build an online version of QDMR heuristic called DQDMR (*Dynamic* QDMR) and an online version of EPDT heuristic DEPDT (*Dynamic* EPDT).

The QDMR, DQDMR, and DEPDT heuristics change the priority of only destination nodes in choosing paths. In the online heuristics we hypothesize that considering either edges or nodes already in the tree for inclusion in subsequent paths would further lower the tree cost. In Section 3.1 we propose tree-node-priority online variants of QDMR and EPDT called TQDMR (*Tree* node priority based online QDMR) and TEPDT (*Tree* node priority based online EPDT). The relationships among DDMC and QDMR related heuristics is illustrated in Figure 2.1.

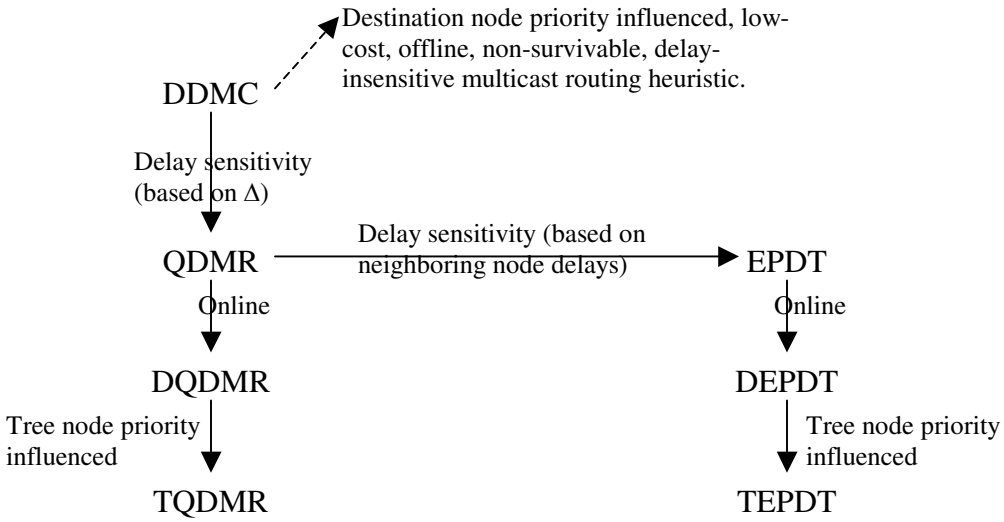

Figure 2.1: QDMR and related heuristics map for non-survivable low-cost delay-constrained multicast routing

2.1.2 Survivable low-cost delay-constrained multicast routing heuristics

Survivable low-cost delay-constrained multicast routing heuristics ensure survival of multicast routing, in the face of failures along a single path, by constructing low-cost multicast graphs having at least two node/edge disjoint paths to each destination while satisfying the delay requirements of the application along each such path. Depending on the approach used in protecting the multicast tree, the heuristics are classified as:

- *Dual Multicast Trees* (DMT): DMT heuristics construct two node/edge-disjoint multicast

trees servicing the different multicast destination nodes. One approach to constructing a node-survivable DMT is presented in [10]. In this approach the internal nodes and edges in the primary tree are removed before constructing the secondary tree, limiting the heuristic's ability to find feasible paths.

- *Single Multicast Graph with Protection* (SMTP): SMTP heuristics construct a single multicast tree. However, each *vulnerable segment* in the tree is protected by providing a back-up path between the end nodes of the segment. A vulnerable segment is a path beginning or ending with the source node, a destination node or an intermediate fork node. An intermediate fork node is one that services destinations on more than one out-going edge. In [12] the authors present an approach for constructing an edge-survivable SMTP. Identification of vulnerable segments and providing back-up paths to each such segment in the tree is a complicated process. A simpler approach would assume the vulnerable segment to be a path from the source node to a destination node and provide a back-up for this segment (or provide a disjoint path pair to each destination in the multicast graph).

- *Multicast Graphs with Disjoint Path Pairs* (MGDPP): This approach constructs a single multicast graph. However, the multicast graph constructed has disjoint path pairs to each multicast destination node. The paths in a path pair might share edges/nodes with other path pairs in the network but do not share edges/nodes with each other.

Construction of survivable point-to-point paths has been investigated by Suurballe in [17] and later by Bhandari in [18]. Bhandari's vertex splitting algorithm [18], Figure 2.2, constructs a node-disjoint shortest path pair from a single source to a single destination in the network. The path pair $(\hat{P}^1_{sd}, \hat{P}^2_{sd})$ constructed is optimal in the sense that $\forall$disjoint path pairs (P^1_{sd}, P^2_{sd}) between s and d:

$$(D_{\hat{P}^1_{sd}} + D_{\hat{P}^2_{sd}}) \leq (D_{P^1_{sd}} + D_{P^2_{sd}})$$

The algorithm consists of two passes of a suitable shortest path algorithm (such as a modified Dijkstra's algorithm, [18], or the Bellman-Ford algorithm, [19]) with a network transformation in between.

After the first pass of the shortest path algorithm the network transformation in Bhandari's algorithm, Figure 2.2, first replaces each edge along the shortest path (found in the first pass) with an arc directed towards the destination. It then splits each internal node along the shortest path into two colocated nodes and joins them with a zero length arc directed towards the destination. Then each external edge originally incident on the internal node is split into two arcs such that one arc emanates from one co-located node and the other terminates on the other co-located node and together with the zero-length arc they form a cycle. The tranformation ends after assigning negative weights to the arcs along the shortest path and reversing the direction of the arcs. Then the second pass of the shortest path alogorithm is run resulting in the second path in the path pair.

In interlacing paths section of the algorithm, any common edges in the paths of the path pair are removed and remaining edges of the path pair are re-grouped to form the final node disjoint shortest path pair. Finally the tranformed graph is restored by uniting the split nodes along the shortest path, removing the zero-length arcs, and transforming the arcs along the shortest path into their original edges.

The paths of a shortest path pair found by Bhandari's algorithm may fail to meet a delay constraint when paths of a different (possibly longer in total) path pair would. Thus, the shortest path pair algorithm is only a heuristic for the delay-constrained problem (an NP-complete problem).

A heuristic for incremental construction of an edge survivable MGDPP called *Optimal Path Pair based Shared Disjoint Paths* (OPP-SDP) algorithm was presented in [12]. In OPP-SDP edge-disjoint path pairs to destination nodes are built sequentially using Suurballe's disjoint path pair algorithm [17] and using the delay experienced on the edge as the metric. In this approach the delay experienced on the edges already present in the tree is set to zero to favor their re-use as each new multicast destination node is added.

BhandariTransform(G, P_{sd})
{
 Replace each edge $e \in P_{sd}$ with an arc a directed towards d
 Split each internal node $n_i \in P_{sd}$ ($n_i \notin \{n_s, n_d\}$) into two co-located
  nodes n_i^1, n_i^2 such that $|a_{n_i^1 \to n_i^2}| = 0$ with n_i^2 closer to d
 Split each external edge e_{xi} and e_{ix} originally incident on internal nodes
  in P_{sd} into two arcs one emanating from $a_{n_i^2 \to n_x}$ and the other
  terminating $a_{n_x \to n_i^1}$ on the co-located nodes such that
  $a_{n_i^1 \to n_i^2}, a_{n_i^2 \to n_x}, a_{n_x \to n_i^1}$ forms a cycle
 Reverse direction and assign negative weights to arcs along P_{sd}
}

InterlacePaths(P'_{sd}, P''_{sd})
{
 (P^a_{sd}, P^b_{sd}) = RemoveCommonEdges(P'_{sd}, P''_{sd})
 (P^1_{sd}, P^2_{sd}) = RegroupRemainingEdges(P^a_{sd}, P^b_{sd})
 Return (P^1_{sd}, P^2_{sd})
}

RestoreTranformedGraph(G, P'_{sd}, P''_{sd})
{
 Unite the split nodes into the original nodes
 Remove the zero length arcs
 Transform the arcs along the shortest path into their orginal edges
}

Figure 2.2: Bhandari's algorithm

OPP-SDP as it favors re-use of edges finds paths more successfully and produces lower cost multicast graphs than approaches such as [10], which construct DMT and compute secondary trees without re-using any edges or nodes from the primary tree. However, since OPP-SDP favors re-use of edges in the multicast tree, the paths discovered are not the shortest ones possible and may, unnecessarily, fail to meet the delay requirements of the application.

2.2 Selection of appropriate graph models for evaluation of routing heuristics

2.2.1 Network Generation Methods

For analysis of a multicast routing heuristic's performance and to predict its behavior in a real-world scenario it is necessary to generate graphs which closely resemble the target networks. There are three major types of graph generation methods in practise [20]: a. *random*, b. *regular*, c. *hierarchical*.

In random methods (due to Erdös and Rényi, [13]) the nodes in the graph are placed randomly on a plane and an edge between any two nodes is added with the help of a probability function. There is no inherent structure in this type of graphs. In regular methods the edges between nodes are added in a structured manner for example rings and meshes. In hierarchical methods larger graphs are recursively built from smaller graph components. The smaller graph components (for example a domain) are connected and are usually built using the random methods.

2.2.2 A quantitative comparison of graph-based models for Internet topology

In [20] Zegura et al. present a methodology for selecting graphs resembling target networks such as the Internet. This methodology is useful for studying problems such as multicast routing through a quantitative comparison of graph generation methods. The prime focus of these comparative studies is on the generation and evaluation of flat random (non-hierarchical) and hierarchical graphs. The flat random methods studied are Pure Random (Random for short) and Waxman, [14], along with the new methods introduced in the paper Exponential and Locality. The hierarchical methods

Table 2.1: Flat Random Graph Generation Methods

Method	Random	Waxman	Exponential	Locality
Probability	p	$\alpha e^{\frac{-d_{uv}}{\beta \times L}}$	$\alpha e^{\frac{-d_{uv}}{(L-d_{uv})}}$	α if $d_{uv} < C$
				β if $d_{uv} \geq C$

considered in the study are N-Level and the Transit-Stub (TS) method.

In the flat random methods the nodes are distributed in a plane and the probability of an edge existing between any two nodes u and v in the plane varies based on the generation method as shown in Table 2.1.

In Table 2.1, p is a fixed number, $0 < \alpha, \beta \leq 1$ are user parameters, d_{uv} is the Euclidean distance between the nodes u and v, L is the maximum distance between any two nodes in the graph, C is the parameter identifying two edge classes for the Locality method with each class acquiring a different probability value (α or β) based on the distance between the two nodes being considered.

The N-Level method constructs a hierarchical graph by iteratively expanding a top-level connected random graph on a square plane. The square is divided into equal-sized square units and the nodes in the graph are assigned to these square units.

During every iteration of the N-Level method each node in the top-level graph is replaced by a lower-level connected graph and the top-level edges are replaced by those connecting the replaced lower-level graphs. The lower-level connected graph is built on a square of size equal to the square unit housing the node it is replacing.

The TS method constructs graphs that resemble the topological properties of the Internet. The Internet is a network of computer networks (or routing domains). A routing domain is either a *stub* or a *transit* domain. Information exchanged between two nodes u and v in the network only passes through a stub domain S if either $u \in S$ or $v \in S$, a restriction differentiating it from a transit domain. Transit domains (or back-bone networks) transfer information between nodes located in lower-level stub domains attached to them.

A TS graph is constructed by first generating a flat connected random graph each node of which is replaced by another connected graph representing a transit domain resulting in a network of transit domains. Each node in every transit domain is then connected to one or more new connected graphs representing the stub domains connected to that node forming a TS graph. If required additional Transit-Stub or Stub-Stub edges are added to improve connectivity.

Several evaluation metrics are used to compare the graph generation methods. The topological metrics considered in the work are:

- *Average node degree*: The ratio of twice the total number of edges over the total number of nodes in the network $\frac{2|N|}{|E|}$.

- *Diameter*: The diameter of a network is the longest path length in the set containing the shortest paths between any two nodes in the network. The shortest paths can be computed and measured based on several metrics such as hop-count, Euclidean edge length, and routing policy weight. Composite diameter metrics such as *hop-length* are also considered. For example, if hop-length is considered as the diameter metric then the shortest paths are computed using the hop-count metric but are measured using the Euclidean edge length metric.

- *Number of biconnected components*: The number of distinct connected sub-graphs in a graph, such that any two edges in each sub-graph are on a common simple cycle. A small number of biconnected components in a graph usually implies that it is well connected.

Application specific metrics considered in the work are (MRA refers to the multicast routing algorithm being evaluated):

- *Packet-hops ratio*: The ratio of the number of edges in all the multicast trees from all sources to all receivers (with a multicast tree constructed for each source) constructed by the MRA to the number of edges in all the multicast trees constructed by an algorithm which constructs

shortest path trees. This metric gives us an idea of the number of hops travelled by the packets.

- *Delay ratio*: The ratio of the maximum delay observed in the MRA to that observed using an algorithm which constructs shortest-path trees.

Comparison of graph generation methods

The evaluation process is rationalized by making the graph generation methods generate graphs with a fixed number of nodes and a fixed average number of edges.

In the hierarchical methods higher level domains exist to connect lower level domains with limited connectivity between the different levels resulting in a higher number of biconnected components than the flat random methods. However connectivity within the domains is better and hence paths have more (usually shorter) intra-domain edges and fewer (usually longer) inter-domain edges resulting in lower length but higher hop diameters than the flat random methods.

A comparison of the hierarchical methods reveals that the TS method, because of the extra transit level, has lower connectivity among the different levels resulting in a higher number of biconnected components than the 2-Level method ($N = 2$). Extra Transit-Stub and Stub-Stub edges will increase connectivity and decrease the number of biconnected components for the TS method.

Lack of a higher level in the 2-level method results in higher policy-length diameter than the TS method but does not affect the policy-hop diameter. The three-tier domain structure of the TS method results in fewer (usually longer) inter domain edges and hence the path lengths are lower resulting in lower policy-length diameters than the 2-Level method.

Multicast Routing

The differences in graph generation methods' performance affecting center-based multicast routing are also explored. Packet-hops and maximum delay are used as metrics for the evaluation. For each run of the experiment the maximum delay and packet-hop values are recorded assuming each node

as the center of the shared tree. This leads to the collection of two results: maximum delay and packet-hops for an optimal center and maximum delay and packet-hops averaged over all centers. These results are normalized as before to those obtained by a shortest-path-tree routing algorithm.

The different experiments conducted in this section help us understand whether the graph generation method's performance has an affect on center-based multicast routing. From the tests it is evident that there is a significant difference in the optimal center ratios[1] of the three methods (Random, Exponential, and TS) with very little difference observed for average center ratios[2]. The means for the optimal center ratios for both packet-hops and maximum delay vary by $12-13\%$ between the two flat-random methods (very little difference between themselves) and the hierachical TS .

However the performance variance observed in the TS method itself (variance between the 25^{th} and 75^{th} percentile) is significantly less than that observed in the flat-random methods. Therefore for an equivalent sample size respresentation one would require a higher number of Random or Exponential graphs than TS graphs for evaluating the performance based on these metrics.

Also the frequency distribution of the metric values (ratios) for the TS method is different to that of the two random methods with most runs (a much higher number than for the flat methods) achieving multicast trees closer to those obtained by a shortest-path-tree routing algorithm (metric value ratio = 1). This frequency distribution is more spread-out in the flat methods. Lower variance in frequency distribution in TS method is because of the topology of the TS graph generated. TS graphs have low connectivity between different domain levels limiting the routing possibilities, hence center based trees constructed are closer to the shortest-path trees for these methods.

These results indicate that the selection of a particular graph generation method for a routing problem is important and an awareness of how different graph generation methods affect a multicast

[1]The ratios of maximum delay and packet-hops for an optimal center in center-based multicast routing to the maximum delay and packet-hops obtained in a shortest-path-tree routing algorithm

[2]The ratios of maximum delay and packet-hops averaged over all the centers in center-based multicast routing to the maximum delay and packet-hops obtained in a shortest-path-tree routing algorithm

routing algorithm's performance is essential for truer and more broader understanding of such an algorithm's performance.

2.2.3 A Better Model for Generating Test Networks

Doar introduces a similar (but different) approach to TS in [21]. This approach henceforth called Layers views a network as a set of networks each of which is made of smaller networks. Layers first creates a top-level network of WANs to which it attaches smaller networks representing the MANs. Nodes in each MAN are then connected to a number of LANs.

The number and size of the WANs, MANs, and LANs is specified by parameter values. LANs in the network are connected using a star topology. In the top two levels of the hierarchy (WAN, MAN) intra-network edges are added with the help of minimum spanning trees.

Researchers responsible for the TS and Layers graph generators stress the importance of hierarchical structure of the Internet. They say that the multi-level domain structure is the most important distinguishing feature affecting routing decisions in such networks. Because of the importance of the hierarchical structure accorded in the topology generation for these methods they are called structural generators [22].

2.2.4 Degree-Based Topology Generators

In 1999 Faloutsos et al. published a seminal paper, [23], which stated that the Internet degree distribution follows a power-law relationship. Thus the graphs generated by the structural generators like TS and Layers were not considered to be adequate representations of real internetworks. As a result degree-based generators which generate graphs with power-law degree distributions have been developed. One such generator; power-law random graph generator (PLRG), [24], assigns degrees drawn from a power-law distribution to the nodes in the graph. In PLRG the number of nodes (n) with degree (d) is:

$$n = \frac{e^{\alpha}}{d^{\beta}} \tag{2.1}$$

where e is the base of the natural logarithm, and α and β are parameters to the graph model. If the degree distribution of the nodes in the graph is drawn on a log-log scale, β is the negative slope while α is the intercept of the degree-distribution line.

PLRG assigns links to the N nodes in the graph by making d_x clones of every node x (d_x is the degree of node x) in the graph. It then randomly assigns edges to the clones of a pair of nodes and repeats this assignment until there are no clones of any node that are not assigned to an edge left. This matching is then translated to the original graph. If an edge exists between a clone of node u and a clone of node v then an edge is added between nodes u and v in the original graph.

Barabási and Albert [25] generate power law topologies using a graph evolution model. The graph evolution model produces networks with either power-law or exponential degree distributions based on the frequency of the following three processes: addition of new nodes to a graph; modification of existing edges in the graph; and addition of new edges to the graph.

The graph evolution model is an extension of the model proposed in [26] for the generation of power law topologies which identified and incorporated in its construction two mechanisms responsible for the existence of power laws in networks such as the Internet. They are: networks grow continuously through the addition of new nodes that attach to existing nodes, and new nodes in the network are more likely to connect to existing higher degree nodes.

2.2.5 Comparison of Structural and Degree-Based Generators

In [22] Tangmunarunkit et al. compare structural generators to degree-based generators using topological metrics. The comparison in these metrics shows that degree-based generators are better at generating graphs which resemble the Internet than are the structural generators. Surprisingly the graphs generated by degree-based generators have a loosely hierarchical structure more similar to the structure of the Internet than the graphs generated by structural generators.

Three different types of networks are used in the comparison process:

- *Generated Networks:* The network graphs are generated using three types of graph generators (representatives of such generators used in the comparison are also shown):
  - *Flat Random Generators:* Waxman method .
  - *Structural Generators:* TS and Layers method.
  - *Degree-Based Generator:* PLRG method.
- *Measured Networks:* Measured networks are graph representations of actual networks such as the Internet built using detailed measurement studies. The measured Internet graphs used in the comparison process are: autonomous system level (AL) graph built using BGP routing tables and the router level (RL) graph built with the help of traceroutes [27].
- *Canonical Networks:* Canonical networks are parametrizable and well established non-real world networks, useful as a distinguising type in the comparison of generated and measured networks. Mesh, k-ary tree, and Erdos-Renyi random graph are the canonical networks used in the comparison process.

The focus in the comparative study is more on illustrating qualitative differences among the different networks by classifying the performance as high or low than showing quantitative differences. To effectively compare such disparate networks and to usefully distinguish them, several topological metrics were investigated. Three of those metrics were found to effectively capture all the distinguishing features of these networks while taking into account differences such as network size (these metrics are then used to distinguish the different types of graph generators):

- *Expansion X(h):* Ratio of the number of nodes which are within h hops from a node in the graph to the total number of nodes in the graph. This is calculated by computing the number of nodes within h hops (X_i^h) from each node n_i in the graph, averaging the result over all nodes in the graph, and dividing the average by the total number of nodes in the graph.

Expansion is an improvement over the un-normalized reachability metric, [28], and is useful in comparison of un-equal size networks.

$$X(h) = \frac{\frac{\sum_{n_i \in N} X_i{}^h}{|N|}}{|N|}$$

- *Resilience R(p):* Average minimum number of links in an p-node ball (ball containing p nodes), centered around a node, that must be cut to create two equal dis-connected partitions of the original graph in the ball [29]. The ability of the graph to remain connected in the face of link failures is illustrated by the Resilience metric. If the minimum number of links in an p-node ball centered around a node n_i that need to be cut to create two equal disconnected partitions of the original graph in the ball are $R_i{}^p$ then:

$$R(p) = \frac{\sum_{n_i \in N} R_i{}^p}{|N|}$$

- *Distortion:* Distortion, [30], measures how a spanning tree S affects routing choices in a graph G. Distortion of S is defined to be the average minimum number of extra edges needed to travel between any two nodes of S which are end-nodes of an edge in G if the travel is restricted to using S. Distortion for the graph is minimum such distortion over all possible spanning trees of the graph. Distortion metric D(n) computes the distortion of a n-node ball centered around a node in the graph. If $D_i{}^p$ is the distortion of a p-node ball centered around node n_i in the graph then,

$$D(p) = \frac{\sum_{n_i \in N} D_i{}^p}{|N|}$$

Both the measured networks exhibit high expansion, high resilience and low distortion. This high-low combination of the three metric values of the measured networks is only exhibited by the

degree-based generator PLRG and not by the structural generators.

The tests were also run on four other degree-based generators (apart from PLRG) all of which produced graphs that had high expansion, high resilience, and low distortion matching the measured networks. Thus degree-based generators are better at producing graphs which are similar to the Internet than the structural generators.

Hierarchy

For an understanding of the closeness of the different networks hierarchical structure to the Internet, a link-based hierarchical metric *the vertex cover of the traversal set of a link in the network* (for brevity called *link importance* in this work)[3] was developed. Link importance of a link is the minimum number of nodes that are dependent and hence affected by the loss of that link in the network. In a flat network all links have similar values for link importance while in a hierarchical network some links (for example links in the backbone network) are used more often than others and hence have higher link importance values than others.

TS, Layers, and Tree construct strictly hierarchical graphs[4]. RL, AL, and PLRG fall into the moderately hierarchical group (here the highest link importance values are lower than those observed in the strictly hierarchical group). Random, Mesh and Waxman have almost uniform link importance distribution suggesting that these networks have little or no hierarchy. Therefore, the moderately hierarchical nature of the Internet is better reflected in degree-based generators than the structural generators which have much stricter hierarchy.

Also, PLRG networks exhibit a high degree of correlation betweeen link importance values and degree of the end nodes of the links. This high level of correlation in PLRG implies that links with high link importance more often than not connect nodes with high degree. Since PLRG follows a power-law degree distribution, there are a significant number of nodes with high node degree. So, a good fraction of links in the network have high link importance values.

[3]Less informative term *link values* is used in [22] for this metric.

[4]Level of hierarchy is classified into 3 groups: strict, moderate, and loose.

Therefore even though degree-based generators enforce no explicit structure in the construction process the power-law nature of the degree distribution ensures a moderate level of hierarchy matching the moderately hierarchical nature of the Internet. In summary, the degree-based generators such as PLRG are better at generating networks with similar characteristics to the Internet.

2.3 Structure and size of multicast destination sets

2.3.1 Measuring and Modelling the Group Membership in the Internet

Cui et al. develop a group membership model in [31] that reflects membership characteristics in real internetworks. Data from real network scenarios such as IETF-Audio, IETF-Video, NASA multicasts over the Mbone and membership characteristics in net games (network games) are used to measure membership metrics and for building a group membership model. The group membership model is useful in generating simulated multicast membership distributions across the network mirroring real distributions and is thus helpful in evaluations of multicast routing protocols. The group membership metrics used in the work are:

- *Member Clustering*: The member clustering metric quantifies the network proximity of the multicast group members. Clustering is calculated using the network-aware clustering method, [32], which first extracts the IP addresses and netmasks of the group members from BGP dump tables and groups all the members with the same longest matching netmask into a cluster.

- *Group Participation Probability*: Probability that a cluster is part of a multicast group.

  - *Multiple Group Participation*: The ratio of the number of multicast groups (g_c) that the cluster c is associated with to the total number of groups (g) in the network: $\frac{g_c}{g}$.

  - *Time-based Participation*: The percentage amount of time that cluster spends in the multicast group.

- *Pairwise correlation in Group Participation*: Probability that two independent clusters are part of a group. Pairwise correlation between two clusters is the correlation coefficient (normalized covariance) of the two clusters.

Knowledge of member clustering can be very useful in multicast routing protocol design and deployment. In a network in which more members are clustered (closer to other members) it is advantageous to have a multicast tree to distribute data among the members. Clustering was observed to be insignificant for members of net games. Only about 10% (maximum of 16 multicast members) of the clusters have 2 or more group members in net games while the corresponding figures for MBone real and cumulative data sets is 20% (50 multicast members) and 60% (500-1000 multicast members) respectively.

Studies conducted on the group participation probability of the clusters show that the probability distribution is not uniform across all scenarios. The group participation probability is not uniform for MBone. The cumulative distribution of participation probabilities for MBone shows a linear increase from 0 to 1. Whereas more than 95% of the clusters in net games have a participation probability of less than 0.1 indicating a more uniform group participation probability.

The pairwise correlation in group participation is observed to be strong in MBone related datasets while being weak in netgames related datasets. This tends to confirm the hypothesis that group membership model reflecting uniform node participation probability is not reflective of all the multicast membership models. Therefore, a new comprehensive group membership model (GEM) is presented.

GEM first utilizes the pre-specified clustering method to assign nodes to different clusters. It then creates groups and randomly associates member clusters to the groups using group participation probability distribution and pairwise correlation in group participation. GEM then selects nodes for these clusters based on the pre-specified cluster size distribution.

Three different desirable member cluster generation algorithms are presented in the paper:

1. *Uniform distribution without correlation*: The probability distribution of clusters joining a multicast group is uniform. Also clusters in the network act independently. This algorithm is useful for modelling net games member distributions.

2. *Non-uniform distribution without correlation*: The probability distribution of clusters joining a multicast group is non-uniform. However clusters in the network still act in an independent manner.

3. *Non-uniform distribution with correlation*: The probability distribution of clusters joining a multicast group is non-uniform. However some pairs of clusters are more likely to be a part of a multicast group. This algorithm is useful for modelling MBone member distributions.

Experimental validation of GEM is also conducted. The plotted curves of the metrics for the multicast groups generated (through GEM) and measured (through orginal datasets) show a good match for group participarion probability distribution and cluster size distribution and a near match for pairwise correlation in group participation.

2.3.2 On Multicast Trees: Structure and Size Estimation

Dolev et al. characterize the structure and size of the multicast trees extracted from the internet in [33]. It is shown that such multicast trees follow power law distributions first reported by [23] for rank-degree and for sub-tree rank-size. Building on an earlier work, [34], which discovered that the distribution of the number of nodes at a distance from a particular node in the Internet is close to a Gamma distribution, the distance distribution of nodes of a particular degree (2 to 6) from the tree root is explored and is found to be close to a Gamma distribution.

Also, most high-degree nodes are found to be located at the core of the network closer to the tree root. The results also show that multicast trees where nodes have a higher node degree (better connected) have a low tree-height. And a linear relationship, both through empirical investigation and analytically, is identified between the number of high-degree nodes of at least a particular

degree and the number of *clients*[5] in the multicast tree. This empirical relationship is established through the use of a new fast sender-based client size estimation algorithm which uses the power law distributions in the network topology to project an estimate of the client-size.

2.4 Evaluation Metrics

In order to judge the quality of the multicast tree constructed it is necessary to identify and define applicable evaluation metrics for evaluating the multicast trees. There are two kinds of evaluation metrics: *application metrics*, and *topological metrics*.

2.4.1 Topological Metrics

The performance of a multicast algorithm is also dependent upon the topology of the networks used. Therefore, it is important to first understand and then evaluate the characteristics of the networks on which the multicast algorithm is run. Some of the topological metrics (used in [20]) are average node-degree, diameter and number of bi-connected components described in the Section 2.2.2:

In [22] three topological metrics useful for distinguishing the graphs generated by structural and degree-based generators were investigated. The three metrics are (Section 2.2.5 contains an explanation of these metrics) : expansion, resilience, and distortion.

In [35] the size and stability of the multicast architecture of the Internet is analyzed. Analysis is based on monitoring data collected over a period of three years from four topologically seperated core routers. A new metric called *Connectedness* of the network is introduced:

- *Connectedness*: Connectedness of a multicast capable network is the total number of multicast capable routers attached to the network.

Any evaluation technique developed should be able to identify evaluation metrics both application and topology specific which are most applicable for the evaluation of a particular multicast

[5]The leaf nodes in the multicast tree are called clients.

routing heuristic.

2.4.2 Application Metrics

Some of the most commonly used application metrics are:

- *Tree cost*: Sum of the costs of all the edges in the multicast tree $\sum_{e \in E_M} c_e$.
- *Blocking Probability*: Probability that a request for multicast tree construction will be blocked for the lack of sufficient resources in the network. We estimate blocking probability through the metric *fraction of connections blocked* (FCB). FCB is the ratio of the number of times the heuristic is successful in constructing a multicast tree (satisfying the requirements of the application such as Δ) to the total number of requests for the construction of multicast tree if each request is made with a different destination set.

  $$FCB = 1 - \frac{\sum_{M \in MM} \text{Sat}\Delta(G, H, M, \Delta)}{|MM|}$$

  where, MM represents the different multicast destination sets.
- *End-to-End Delay*: Maximum of the delays from the source to the multicast destination nodes $\max_{d \in M} D_{P_{sd}}$.

Evaluation metrics for IP multicast protocols are presented in [36]:

- *Tree Construction Latency*: The time required to construct a multicast tree spanning the multicast destination set.
- *Member Join Latency*: The time elapsed between the issuance of a join-request by a new member and data reception at the new member node.
- *Member Leave Latency*: The time elapsed between the issuance of a leave-request by a member and the end of data reception at the member node.

- *Transmission Latency*: Time taken to transmit a packet from the source to the multicast destination node. This metric is the same as to the end-to-end delay metric defined earlier.

- *Delay Jitter*: The difference between the maximum and minimum end-to-end delays experienced at a node.

- *Aggregate Bandwidth Consumption*: Sum of individual bandwidth consumption of all links in the multicast tree. Individual bandwidth consumption of a link is the bandwidth utilized at that link in support of the multicast tree.

- *Delay Variation*: The delay variation between any two nodes is the absolute difference between the end-to-end delays experienced by the two nodes. Maximum inter-destination delay variation is the maximum difference between the end-to-end delays experienced by any two destination nodes. A multicast routing algorithm satisfying delay and *delay variation* constraints is presented in [15]:

- *Available Bandwidth*: Bandwidth available on the link after the allocation of requisite bandwidth to the multicast tree, if the link is part of the tree. A new multicast routing algorithm in [37] attempts to maximize *available bandwidth* while satisfying delay and delay variation constraints.

In [38], approaches for the creation of a set of multicast trees are presented. The idea behind maintaining a set of trees is that in case the first multicast tree fails it can be replaced by an alternate tree to minimize the interruption time for the multicast routing. Several new metrics are also presented in this work:

- *Average Cost of a Multicast Scheme*: The weighted average of the tree cost all the trees computed by the scheme (weighted by the fraction of time each such tree is used).

- *Dcost of a Dtree*: A *Dtree* is a sub-graph of a multicast tree composed of a parent node (any non-leaf node in the multicast tree) and its child edges such that this node and its edges do

not appear in any other Dtree of the multicast tree. The *Dcost* of a Dtree is the cost of an edge in the Dtree whose cost is the maximum among all the Dtree edges. The Dcost of a multicast tree is the sum of the Dcosts of all the Dtrees in the multicast tree. This metric is applicable to ad hoc mobile networks where the transmission cost from a node to other nodes in its vicinity is equivalent to the cost of transmission to the highest cost node.

- *Dcost of a multicast tree*: The Dcost of a multicast tree is the sum of the Dcosts of all the Dtrees in the multicast tree.

- *Average Dcost of a Multicast Scheme*: Average Dcost of a multicast scheme is the weighted average of the Dcost of all the multicast trees computed by the scheme (weighted by the amount of the time each such tree is used).

- *Time of Failure of the Tree*: Time at which the first failure of a link in the tree occurs.

- *Time of Failure of the System*: Time at which the first failure of all the multicast tree paths to a destination node occurs. Time of failure of the system is also referred to as mean time between interruptions.

- *Increase in Mean Time between Interruptions*: This is the difference between the time of the failure of the system and the time of failure of the tree. Increase in mean time between interruptions is the extra amount of time the tree survives after the first failure of a link in a multicast tree because of the additional multicast trees with alternate paths computed which would not have been possible with only a single tree.

- *Probability of Usefulness*: Ratio of the number of times in which the system time is greater than the time of failure of the tree over the total number of times the tree is run in the scheme.

2.5 Comparison to the best achievable solution

Computing the best achievable solution for the problem space may not always be feasible because of the fact that delay-constrained multicast routing is NP-complete as is the delay constrained disjoint path pairs problem. However in [39] an approach which uses enumeration of sub-graphs for computing best achievable solutions for small networks is provided.

The idea behind enumeration of subgraphs is simple: for each subset of the graphs edges (in a 16 edge graph there can be 65536 such subsets) determine if it is a solution (i.e. determine whether it connects the source to all the destinations in the tree). Determine whether the delay bound requirement to all the destinations is satisfied by the paths to the destinations in the graph. If the delay bound requirement is satisfied then the subset represents a successful candidate multicast tree for the problem space. Compute all such multicast trees and select one which has the lowest achievable multicast tree cost. Also a filtration method is employed to reduce the exploration effort in the enumeration process.

CHAPTER THREE

LOW-COST DELAY-CONSTRAINED MULTICAST ROUTING HEURISTICS

Building on the previous work in low-cost, delay-constrained multicast routing heuristics, this chapter presents a description of all the online, low-cost, delay-constrained multicast routing heuristics (both the edge-priority based heuristics proposed in this work and the node-priority based heuristics derived from QDMR) considered in this work.

3.1 Non-survivable low-cost delay-constrained multicast routing heuristics

DWH and the other non-survivable, online, low-cost, delay-constrained multicast routing heuristics considered in this work:

- *First Phase:* Construct delay influenced least-cost (lowest-effective-cost) paths to each destination.
- *Second Phase:* Merge the least-delay paths, if paths constructed in the first phase fail to meet the delay requirements of the application.

In all the heuristics, destinations and edges along the paths to the destination are added incrementally (for each destination) to the multicast tree.

In DWH, edges not present in the multicast tree are assigned a weight coefficient of unity. Edges that are already present in the multicast tree are assigned a weight coefficient based on how close the delay to the end of the edge (from the source) is to a user-specified parameter, k. Tree edges that are closer to violating the delay bound receive a higher coefficient and hence are less likely to be included in subsequent paths.

Using delay-influenced effective cost instead of actual cost of the edge in the path computation means that DWH favors edges already present in the multicast tree for use in paths to new nodes

Table 3.1: Weight Coefficient Calculation

$W_{e_{pq}} = 1$	$(e_{pq} \notin G_M)$
$W_{e_{pq}} = 1$	$(e_{pq} \in G_M \;\&\&\; r_{pq} \geq 1)$
$W_{e_{pq}} = r_{pq}$	$(e_{pq} \in G_M \;\&\&\; r_{pq} < 1)$

thus reducing the tree cost. The weight coefficient for each edge (e_{pq}) in the network is determined by first computing a coefficient, (r_{pq}):

$$r_{pq} = \frac{d_{n_p} + d_{e_{pq}}}{k}$$

The weight coefficients ($W_{e_{pq}}$) assigned to edges are described in Table 3.1.

Two extreme variants of DWH: UWH and ZWH are also considered and evaluated along with the other heuristics. In UWH the edges in the tree retain their original cost values ($W_{e_{pq}} = 1 \; \forall e_{pq} \in G_M$) and hence paths to destinations are the least-cost paths possible, however the tree produced usually has a higher cost than DWH and ZWH. This is because the least-cost paths to some destinations might not meet the delay bound of the application requiring the computation and merging (in to the tree) of the least-delay paths to such destinations in the second phase of the heuristic and thus increasing the multicast tree cost.

In ZWH tree edges are assigned a weight coefficient of zero so that they are more likely to be re-used in paths to other destinations. ZWH favors re-use of edges, but ignores the delay requirements (like UWH) in the first phase, so may sometimes end up with a more expensive multicast tree as it has to merge the least-delay path in the second phase.

In addition to the routing heuristics explained earlier and as noted in Section 2.1.1 four additional online heuristics DQDMR, DEPDT, TQDMR, TEPDT (all variants of QDMR) have been compared in the evaluation. In DQDMR, DEPDT, TQDMR, and TEPDT (like in QDMR) the weight coefficients are assigned to nodes instead of the edges during the computation of the lowest-effective-cost paths. The motivation for these heuristics is as follows:

- *DQDMR* : DQDMR, dynamic QDMR, is the destination node priority influenced online least-cost-path variant of QDMR. In DQDMR the weight coefficient, r_{pq}, assigned to a destination node is:
$$r_{pq} = \begin{cases} \frac{d_{n_p}}{\Delta} & \text{if } p \in M; \\ 1 & \text{otherwise.} \end{cases}$$

- *DEPDT* : DEPDT, dynamic EPDT, is the destination node priority influenced online least-cost-path variant of EPDT. In DEPDT the weight coefficient, r_{pq}, assigned to a destination node is:
$$r_{pq} = \begin{cases} \frac{d_{n_p}}{d_{n_p}+d_{n_q}} & \text{if } p \in M; \\ 1 & \text{otherwise.} \end{cases}$$
In DEPDT the weight coefficient assigned to a destination node is always ≤ 0.5, this is because $d_{n_q} \geq d_{n_p}$. In contrast, DQDMR does not assign such bounded coefficients. DQDMR assigns a weight coefficient of 1 if the node is not a destination node, otherwise the weight coefficient is dependant on d_{n_p} (when p is a destination node).

- *TQDMR*: TQDMR assigns variable priority to the multicast tree nodes instead of just the destination nodes (as done in DQDMR) and hence achieves a better sharing of edges (and hence lower total cost) as the multicast tree nodes are more numerous than the destination nodes. The evaluations show that TQDMR usually constructs multicast trees with lower tree cost than DQDMR. In TQDMR the weight coefficient, r_{pq}, assigned to a tree node is:
$$r_{pq} = \begin{cases} \frac{d_{n_p}}{\Delta} & \text{if } p \in N_M; \\ 1 & \text{otherwise.} \end{cases}$$

- *TEPDT* : TEPDT, a variant of DEPDT, assigns variable priority to the multicast tree nodes (as in TQDMR). The evaluations show that TEPDT almost always constructs multicast trees with lower cost than DEPDT. In TEPDT the weight coefficient, r_{pq}, assigned to a tree node is:

$$r_{pq} = \begin{cases} \frac{d_{np}}{d_{np}+d_{nq}} & \text{if } p \in N_M; \\ 1 & \text{otherwise.} \end{cases}$$

Figure 3.4 presents an algorithmic description of the construction of online, non-survivable, low-cost, delay-constrained multicast tree heuristics considered in this work.

In these heuristics, if the lowest-effective-cost path computed satisfies the delay bound requirement of the application then the nodes and the edges in this path are added to the multicast tree G_M and the next destination node from the destination set is considered for incremental expansion of the multicast tree. This process continues until paths to all the destination nodes are computed at which point the heuristic is deemed to be successful in the construction of the multicast tree for the destination set.

If the lowest-effective-cost path computed fails to meet the delay bound requirement of the application then the least-delay path to the destination node is computed as shown in Figure 3.3. If this least-delay path satisfies the delay bound requirement of the application then the nodes and edges along the path are *merged* to the multicast tree. The multicast tree construction then continues to the next destination node in the destination set.

The lowest-effective-cost path for DQDMR, DEPDT, TQDMR, and TEPDT heuristics is computed as shown in Figure 3.1 by assigning variable priority to certain nodes in the graph. Variable priority is assigned to destination nodes in DQDMR and DEPDT, and multicast tree nodes (including the destination nodes) in TQDMR and TEPDT. The lowest-effective-cost path for DWH, UWH, ZWH heuristics is computed as shown in Figure 3.2 by assigning variable priority to edges in the multicast tree.

Merging is accomplished by adding nodes and edges along the reverse path from the destination towards the source until a multicast tree node v is encountered. If the sum of the cumulative delays to v (from the destination node along the reverse path) and the delay from the source to v (along the forward path from the source) is less than the delay bound requirement, the merging process terminates.

DijkstraQDMR(G, M, G_M, H, n_p, n_d, N^D)
{
  if $n_p = n_d$
   return
  for each e_{pq} {
   if $d_{n_p} + d_{e_{pq}} < \Delta$ {
    if $c_{n_q} > c_{n_p} * \text{CoeffQ}(H, M, N_M) + c_{e_{pq}}$ {
     $c_{n_q} = c_{n_p} * \text{CoeffQ}(H, M, N_M) + c_{e_{pq}}$ {
     $d_{n_q} = d_{n_p} + d_{e_{pq}}$
    }
   }
  }
  n_i = ExtractMinCostNode(N^D)
  DijkstraQDMR($G, M, G_M, H, n_i, n_d, N^D$)
}

CoeffQ(H, M, N_M)
{
  if H = DQDMR
   $r_{pq} = \begin{cases} \frac{d_{n_p}}{\Delta} & \text{if } p \in M; \\ 1 & \text{otherwise.} \end{cases}$
  else if H = DEPDT
   $r_{pq} = \begin{cases} \frac{d_{n_p}}{d_{n_p}+d_{n_q}} & \text{if } p \in M; \\ 1 & \text{otherwise.} \end{cases}$
  else if H = TQDMR
   $r_{pq} = \begin{cases} \frac{d_{n_p}}{\Delta} & \text{if } p \in N_M; \\ 1 & \text{otherwise.} \end{cases}$
  else if H = TEPDT
   $r_{pq} = \begin{cases} \frac{d_{n_p}}{d_{n_p}+d_{n_q}} & \text{if } p \in N_M; \\ 1 & \text{otherwise.} \end{cases}$
  return r_{pq}
}

Figure 3.1: DijkstraQDMR for (DQDMR, DEPDT, TQDMR, TEPDT)

If however the least-delay path fails to satisfy the delay bound requirement then the heuristics are deemed unable to construct the multicast tree for this destination set and the request for the construction of a multicast tree is called blocked.

3.1.1 Computational Complexity

The computational complexity of all the heuristics considered in this section is $O(|M||E|log|N|)$. This is because the Dijkstra algorithm (complexity $O(|E|log|N|)$, [9]) is run once for each node in the destination set. This uniform computational complexity of the heuristics allows for a fair evaluation of the performance of the heuristics.

3.2 Survivable low-cost delay-constrained multicast routing heuristics

Survivable online low-cost delay-constrained multicast routing heuristics incrementally construct low-cost shared disjoint path pairs to each destination while satisfying the delay requirements of the application along both the paths in each path pair. OPP-SDP, [12], constructs low-cost shared

```
DijkstraDWH(G, M, G_M, H, n_p, n_d, N^D)
{
  if n_p = n_d
    return
  for each e_pq {
    if c_{n_q} > c_{n_p} + c_{e_pq} * CoeffD(H, E_M) {
      c_{n_q} = c_{n_p} + c_{e_pq} * CoeffD(H, E_M)
      d_{n_q} = d_{n_p} + d_{e_pq}
    }
  }
  n_i = ExtractMinCostNode(N^D)
  DijkstraDWH(G, M, G_M, H, n_i, n_d, N^D)
}
```

```
CoeffD(H, M, N_M)
{
  if H = DWH
    if e_pq ∈ E_M
      r_pq = { (d_{n_p} + d_{e_pq}) / Δ   if (d_{n_p} + d_{e_pq}) / Δ < 1;
             { 1                          otherwise.
    else
      r_pq = 1
  else if H = UWH
      r_pq = 1
  else if H = ZWH
      if e_pq ∈ E_M
        r_pq = 0
      else
        r_pq = 1
}
```

Figure 3.2: DijkstraDWH for (DWH, UWH, ZWH)

```
DijkstraDelay(G, M, G_M, H, n_p, n_d, N^D)
{
  if n_p = n_d
    return
  for each e_pq {
    if d_{n_q} > d_{n_p} + d_{e_pq} {
      d_{n_q} = d_{n_p} + d_{e_pq}
    }
  }
  n_i = ExtractMinDelayNode(N^D)
  DijkstraDWH(G, M, G_M, H, n_i, n_d, N^D)
}
```

Figure 3.3: DijkstraDelay ($\forall H$)

ConstructLCLDMT(G, H, M, s, N_M)
{
 for each $d \in M$
 {
  $P_{sd} = \begin{cases} \text{DijkstraQDMR}(G, M, G_M, H, n_s, n_d, N - n_s) & \text{if } H \in DQDMR, DEPDT, TQDMR, TEPDT; \\ \text{DijkstraDWH}(G, M, G_M, H, n_s, n_d, N - n_s) & \text{if } H \in DWH, ZWH, UWH. \end{cases}$
  if SatΔ(d) = 1
   Add each $e \in P_{sd}$ and each $n \in P_{sd}$ to G_M
  else
   P_{sd} = DijkstraDelay$(G, M, G_M, H, n_s, n_d, N - n_s)$
   if Sat$\Delta(P_{sd})$
    Merge(G_M, P_{sd})
   else
    return Failure
 }
 return Success
}

Figure 3.4: Low-cost delay-constrained multicast tree

disjoint path pairs to each destination in the graph. However OPP-SDP does not consider the delay requirements of the application in its multicast graph construction.

One approach to improving the delay performance of OPP-SDP is to modify the heuristic so that it uses cost of the edge as its metric in its initial disjoint path pair construction[1]. If the path pair constructed using cost as the metric fails to meet the delay bound of the application then the least-delay path pair from the source to the destination, if found to satisfy the delay bound of the application is added to the multicast graph.

This will ensure that the heuristic will meet the delay bound of the application, if possible, while constructing low-cost multicast graphs. However, since optimal disjoint path pair construction is NP-complete the heuristic may sometimes fail to construct a successful path pair (meeting the delay requirements) even when a feasible solution exists for the problem.

Also redundancy is further improved by using Bhandari's node-disjoint shortest path pair algorithm instead of edge-disjoint shortest path pairs algorithm used by OPP-SDP. This improved heuristic is called the Zero Weight Disjoint Path Pair (ZW-DPP).

Another variant of ZW-DPP proposed and evaluated is the Unit Weight Disjoint Path Pair (UW-DPP) heuristic. In UW-DPP, edges retain their original cost values throughout the multicast graph construction process resulting in least-cost path pairs. However, since edge re-use is not favored the multicast graph cost is usually higher than the cost of graphs produced by ZW-DPP.

An ideal solution would minimize graph cost while meeting the delay requirements. The idea presented in QDMR, [9], for constructing a non-survivable delay-constrained low-cost multicast graph is again used here (as in DWH) with some improvements. This new approach called Dynamic Weight Disjoint Path Pairs (DW-DPP) heuristic dynamically assigns a weight coefficient, $W_{e_{pq}}$, to each edge in the network. The shortest path pair algorithm is run using $W_{e_{pq}} \times c_{e_{pq}}$, the effective cost, instead of $c_{e_{pq}}$ the cost of the edge.

[1]The effective cost of the edges already present in the multicast graph is set to zero to favor their re-use in subsequent path pair calculations

Using effective cost instead of actual cost in the first phase of computation means that DW-DPP favors edges already present in the multicast graph for use in paths to new nodes, but not as aggressively as does ZW-DPP thus reducing the likelihood that the more costly least-delay path pair will be added in the second phase.

As in ZW-DPP and UW-DPP, DW-DPP adds destinations incrementally. Edges not present in the multicast graph are assigned a weight coefficient of unity. And edges that are already present in the multicast graph are assigned a weight coefficient based on how close the delay to the end of the edge (from the source) is to a user-specified parameter, k (as in DWH). The relationships among DWH, ZWH, UWH, and their survivable variants is illustrated in Figure 3.5

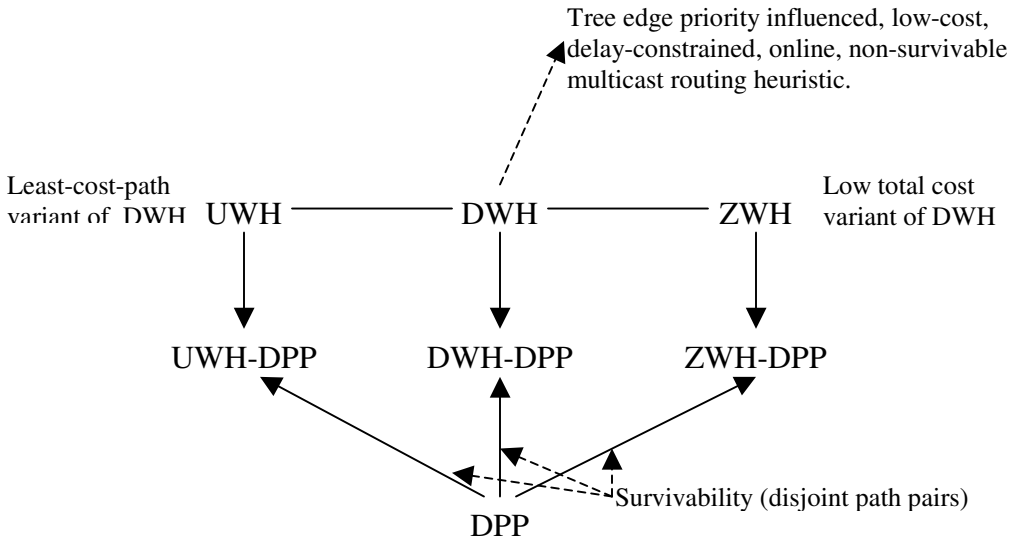

Figure 3.5: DWH and related heuristics map for survivable low-cost delay-constrained multicast routing

As has been done in the survivable case the heuristics DW-DPP, UW-DPP, and ZW-DPP are compared to the disjoint path pair variants of the QDMR based heuristics which are listed below:

- *DQDMR-DPP:* Disjoint path pairs variant of DQDMR.
- *TQDMR-DPP:* Disjoint path pairs variant of TQDMR.
- *DEPDT-DPP:* Disjoint path pairs variant of DEPDT.

- *TEPDT-DPP:* Disjoint path pairs variant of TEPDT.

The relationships among the QDMR based heuristics and their survivable variants is illustrated in Figure 3.6.

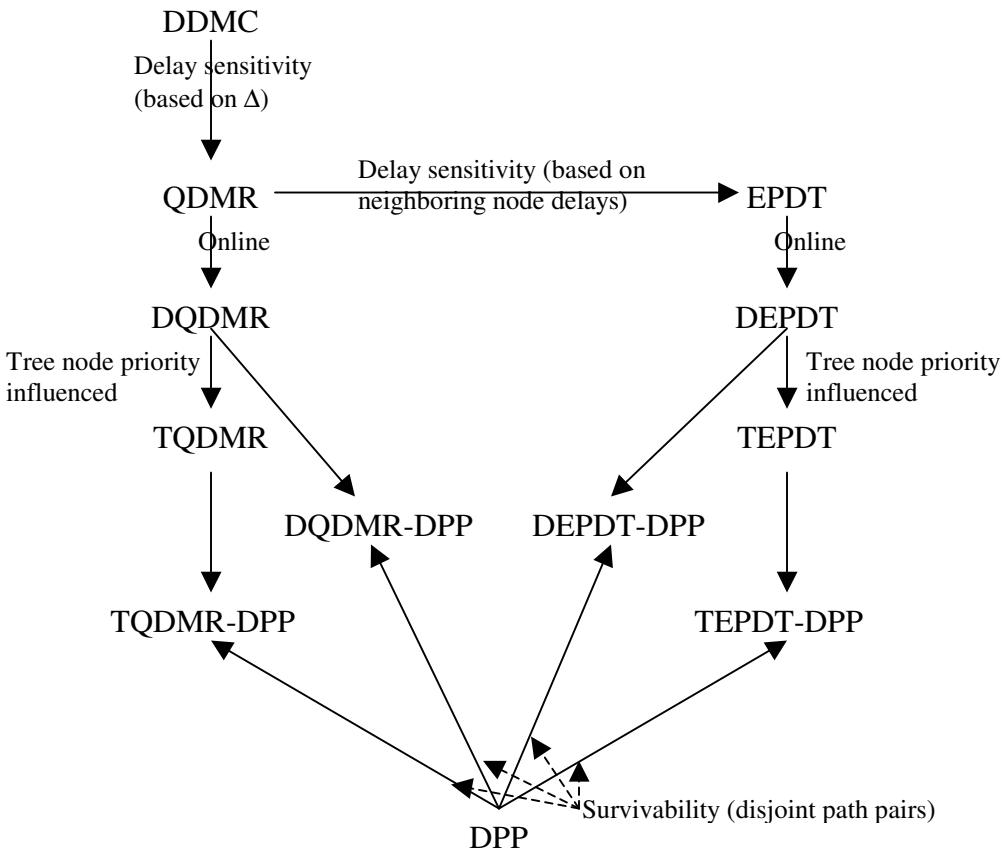

Figure 3.6: QDMR and related heuristics map for survivable low-cost delay-constrained multicast routing

The capabilities of all the multicast routing heuristics considered in this work are illustrated in the Figure 3.7. The capabilities listed in the columns convey whether the particular multicast routing heuristic:

- Incrementally constructs multicast graphs for each new destination.
- Constructs low-cost multicast graphs.

- Constructs multicast graphs with delays on the paths from the source to destination nodes meeting the delay bound.
- Builds shared disjoint path pair multicast graphs.
- Constructs multicast graphs by assigning varying priorities to nodes in the graph.
- Constructs multicast graphs by assigning varying priorities to edges in the graph.

Heuristics	Online	Low-cost	Delay-constrained	DPP	Node-priority	Edge-priority
DDMC	–	✓	–	–	✓	–
QDMR	–	✓	✓	–	✓	–
EPDT	–	✓	✓	–	✓	–
DQDMR	✓	✓	✓	–	✓	–
DEPDT	✓	✓	✓	–	✓	–
TQDMR	✓	✓	✓	–	✓	–
TEPDT	✓	✓	✓	–	✓	–
ZWH	✓	✓	✓	–	–	✓
UWH	✓	✓	✓	–	–	✓
DWH	✓	✓	✓	–	–	✓
DQDMR-DPP	✓	✓	✓	✓	✓	–
DEPDT-DPP	✓	✓	✓	✓	✓	–
TQDMR-DPP	✓	✓	✓	✓	✓	–
TEPDT-DPP	✓	✓	✓	✓	✓	–
ZW-DPP	✓	✓	✓	✓	–	✓
UW-DPP	✓	✓	✓	✓	–	✓
DW-DPP	✓	✓	✓	✓	–	✓

Figure 3.7: Heuristic Capabilities

Figure 3.9 presents an algorithmic description of the online, survivable, low-cost, delay-constrained multicast graph heuristics considered in this work. In the first phase the heuristics all compute a lowest-effective-cost node-disjoint path pair to each destination using Bhandari's algorithm [18], 2.2.

The lowest-effective-cost path pair is constructed by first computing a lowest-effective-cost path (first path) using the appropriate (for the heuristic) dijstra algorithm, applying the Bhandari transformation and then computing the second path. The first and second paths are interlaced to remove the common edges and the remaining edges are regrouped to form the lowest-effective-cost disjoint path pair.

If both the paths in lowest-effective-cost disjoint path pair satisfy the delay bound requirement, the nodes and edges along the path pair are added to the disjoint path pair multicast routing solution and the heuristic considers the next destination in the destination set. This process continues until path pairs to all the destination nodes are computed at which point the heuristic is deemed to be successful in the construction of a disjoint path pair multicast routing solution for the destination set.

If any of the paths in the lowest-effective-cost path pairs computed fails to meet the delay bound requirement of the application then the least-delay path pair to the destination node is computed. If both the paths in the least-delay path satisifies the delay bound requirement of the application then the nodes and the edges along the path pair are added to the disjoint path pair solution. This continues until there are no destination nodes in the destination set.

If however any of the paths in the least-delay path pair fails to satisfy the delay bound requirement then the heuristics are deemed unable to construct the disjoint path pair multicast routing solution for this destination set and the routing request is blocked.

3.2.1 Example showing the working of DW-DPP heuristic

Figure 3.8(a) shows a 10 node 16 edge graph used for explaining the working of DW-DPP heuristic. In the example graph, the cost and delay values are labeled on each edge as c_e/d_e. For example, for the edge e_{56} the cost and delay values are 5 and 4 respectively. In this example, $\Delta = 15$ and $k = 30$. Here, DW-DPP is trying to construct shared DPP graph spanning destination nodes 5, 8 and the source node 0. DW-DPP incrementally constructs disjoint path pairs to each destination:

Destination 5

In the first phase DW-DPP computes the lowest-effective-cost path pair to destination node 5. As there no edges in the shared DPP graph to start with, DW-DPP computes the least-cost path pair to 5. First path P^a_{05} computed is 0 => 6 => 5. DW-DPP then transforms the graph using Bhandari's technique as shown[2] in Figure 3.8(b) and computes the second path in the path pair, P^b_{05} (0 => 4 => 5). As there are no interlacing edges in the path pair and as ($D_{P^a_{05}} = 7$) < Δ and ($D_{P^b_{05}} = 10$) < Δ, DW-DPP is successful in finding the path pair for this destination which is added to the DPP graph.

Destination 8

As edges e_{06}, e_{04}, e_{65}, and e_{45} are in the DPP graph, the effective cost on these edges is considered in the first phase of computing the path pairs. For example, effective costs on the edges are:

$$F_{e_{06}} = \frac{d_{n_0}+d_{e_{06}}}{k} \times c_{e_{05}} = \frac{0+3}{30} \times 3 = 0.3$$

$$F_{e_{04}} = \frac{d_{n_0}+d_{e_{04}}}{k} \times c_{e_{04}} = \frac{0+4}{30} \times 4 = 0.53$$

$$F_{e_{65}} = \frac{d_{n_6}+d_{e_{65}}}{k} \times c_{e_{65}} = \frac{3+4}{30} \times 5 = 1.17$$

$$F_{e_{45}} = \frac{d_{n_4}+d_{e_{45}}}{k} \times c_{e_{45}} = \frac{4+6}{30} \times 6 = 2$$

Figure 3.8(c) shows the DPP graph after the edges in the path pair to destination node 5 are added to it. The edges present in the multicast graph are shown in bold and are now labeled with their effective edge costs which will be used in computing path pairs to destination node 8. The lowest-effective-cost path pair to 8 computed using the above effective costs is:

(P^a_{08}, P^b_{08}) = (0 => 6 => 8, 0 => 4 => 5 => 8)

However as ($D_{P^b_{08}} = 18$) > Δ, DW-DPP reverts to the second phase and computes the least-delay

[2]The affected edges are shown in bold. Also note that the split nodes, 6^1 and 6^2 (split from the node 6), are joined by a zero length arc.

path pair to 8.

(P^c_{08}, P^d_{08}) = (0 => 6 => 8, 0 => 3 => 2 => 8)

As $(D_{P^c_{08}} = 7) < \Delta$ and $(D_{P^d_{08}} = 13) < \Delta$, DW-DPP is successful in finding path pairs to node 8 which are added to the DPP graph. Thus DW-DPP successfully finds a survivable, delay-constrained, low-cost multicast graph spanning the destination nodes. Figure 3.8(d) shows the final DPP graph containing all the edges (shown in bold) in the path pair to destination nodes 5 and 8.

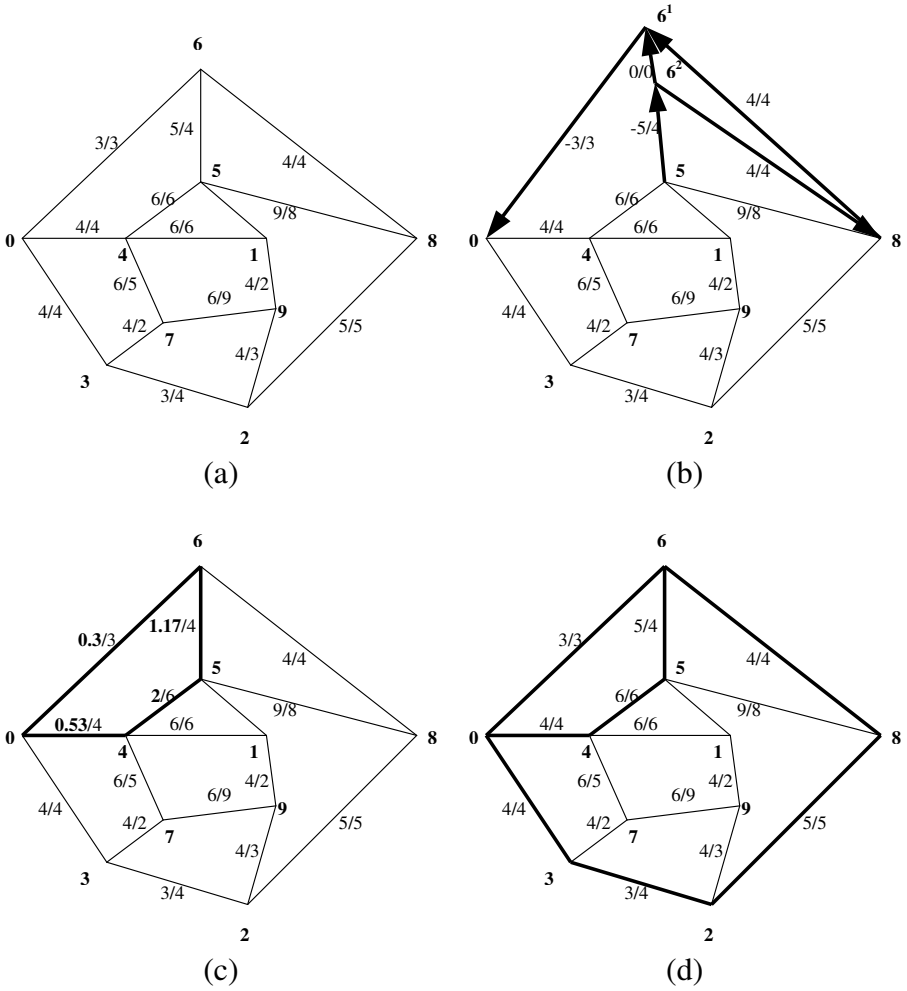

Figure 3.8: A 10 node 16 edge graph for explaining the working of DW-DPP

3.2.2 Computational Complexity

The calculation of running time of shared disjoint path pairs heuristics used in this work is broken down below:

- *First pass of modified Dijkstra's algorithm*: The first pass of the modified dijsktra's algorithm takes $O(|N|^2)$ time.
- *Bhandari's Transformation*: Bhandari's transformation is achieved in $O(|E||N|)$ time.
- *Second pass of modified Dijkstra's algorithm*: The second pass of the modified dijsktra's algorithm takes $O(|N|^2)$ time.
- *Bhandari's Restoration*: Bhandari's method for restoring the transformed graph takes $O(|E||N|)$ time.

The shared disjoint path pairs are computed for each destination in the destination set. Hence the computational complexity of survivable low-cost, delay-constrained multicast routing heuristics is $O(|M|(N^2 + |E||N|))$.

3.3 Evaluation of multicast routing heuristics

Chapter 4 presents an evaluation of the non-survivable low-cost delay-constrained multicast routing heuristics and Chapter 5 presents an evaluation of the survivable heuristics. Both the non-survivable and survivable multicast routing heuristics are evaluated in a structured manner as follows:

1. The heuristics' performance is evaluated on four different Waxman, [14], graphs with varying node degree values. The Waxman method is widely used, [15], [9], [40], [22], and [10] for the generation of random graphs.
2. The heuristics' performance is evaluated and their performance compared on Waxman graphs, a pure random graph (a random graph generation method due to Erdös and Rényi, [13]), a

ConstructLCLDDPP(G, H, M, s, N_M)
{
 for each $d \in M$
 {
  $P^a_{sd} = \begin{cases} \text{DijkstraQDMR}(G, M, G_M, H, n_s, n_d, N - n_s) \\ \quad \text{if } H \in DQDMR, DEPDT, TQDMR, TEPDT; \\ \text{DijkstraDWH}(G, M, G_M, H, n_s, n_d, N - n_s) \\ \quad \text{if } H \in DWH, ZWH, UWH. \end{cases}$
  BhandariTransform(G, P^a_{sd})
  $P^d_{sd} = \begin{cases} \text{DijkstraQDMR}(G, M, G_M, H, n_s, n_d, N - n_s) \\ \quad \text{if } H \in DQDMR, DEPDT, TQDMR, TEPDT; \\ \text{DijkstraDWH}(G, M, G_M, H, n_s, n_d, N - n_s) \\ \quad \text{if } H \in DWH, ZWH, UWH. \end{cases}$
  (P^1_{sd}, P^2_{sd}) = InterlacePaths(P^a_{sd}, P^b_{sd})
  Restore the transformed graph(G, P^a_{sd}, P^b_{sd})
  if Sat$\Delta(P^1_{sd})$ && Sat$\Delta(P^2_{sd})$
   Add each $e \in P^1_{sd}, P^2_{sd}$ and each $n \in P^1_{sd}, P^2_{sd}$ to G_M
  else {
   P^a_{sd} = DijkstraDelay$(G, M, G_M, H, n_s, n_d, N - n_s)$
   BhandariTransform(G, P^a_{sd})
   P^b_{sd} = DijkstraDelay$(G, M, G_M, H, n_s, n_d, N - n_s)$
   (P^1_{sd}, P^2_{sd}) = InterlacePaths(P^a_{sd}, P^b_{sd})
   RestoreTransformedGraph(G, P^a_{sd}, P^b_{sd})
   if Sat$\Delta(P^1_{sd})$ && Sat$\Delta(P^2_{sd})$
    Add each $e \in P^1_{sd}, P^2_{sd}$ and each $n \in P^1_{sd}, P^2_{sd}$ to G_M
   else
    return Failure
  }
 }
 return Success
}

Figure 3.9: Low-cost delay-constrained disjoint path pair heuristics

locality graph (a graph generation similar to the Waxman method) proposed by Zegura in [20], and a hierarchical Transit-Stub graph [20].

3. The heuristics' performance is evaluated on 100 different Waxman graphs.

4. The heuristics' performance is compared to the optimal solution which is obtained by exhaustively enumerating edge subsets on a small graph.

5. The evaluations of the heuristics' performance is conducted at varying delay bound values reflecting a wide range of delay requirements of the application.

The evaluations show that the new heuristics, DWH and DW-DPP, at high k generally construct multicast graphs with lower cost than the other online heuristics. Also in DWH and DW-DPP the cost of the multicast graphs constructed decreases with increasing k.

CHAPTER FOUR

EVALUATION OF LOW-COST DELAY-CONSTRAINED MULTICAST ROUTING HEURISTICS

This chapter presents the evaluation of the multicast routing heuristics (QDMR, DQDMR, DEPDT, TQDMR, TEPDT, ZWH, UWH, and DWH) which construct multicast trees with low tree cost while meeting the delay requirements on paths to destinations. Hence the relevant evaluation metric average tree cost is selected to compare the performance of the heuristics at various delay bound values. Also, we are trying to anwser the following questions in our evaluations:

- Is there a change in the heuristic's performance with respect to others in its problem space with varying node degrees in a graph?
- Does the heuristic's performance change when it is executed on different types of graph?
- How is the average total cost performance of a heuristic when it is executed on a large number of graphs?
- How do the heuristic performances compare to the optimal solution on a graph?
- Does the relationship between the cost and delay of an edge affect the performance of the heuristic?
- How does the heuristic's performance vary with varying delay bound values representing varying delay requirements of the application?

Most heuristic evaluations in the literature try to answer a few of the above listed questions by performing the evaluations on a single graph or a few graphs, not considering the different delay cost relationships of the edges, not evaluating the heuristics on different types of graphs, and not comparing the heuristics performance to the optimal solution [11], [10], [12]. A structured evaluation of the heuristics as presented in this work will serve as a useful guide to future researchers.

The evaluation of the heuristics presented in this dissertation covers all of the above mentioned requirements for an effective evaluation of low-cost, delay-constrained multicast routing heuristics. The evaluation process is broken down into the following sections so as to allow for an extensive comparison of the heuristics:

- *Evaluation of the heuristics on 4 Waxman graphs:* In Section 4.1 the heuristics are all compared on 4 different Waxman graphs with average node degrees (6.02, 5.00, 4.04, and 3.04) covering a range of graphs of the same type with varying connectivity. The average node degrees in this set were selected to cover a range of values (with a difference in node degree of about one between adjacent elements) and from a high close to 6 (representing a well-connected graph) to a low of 3.04 (representing a sparsely-connected graph).

- *Evaluation on disparate graphs:* In Section 4.2 the heuristics are compared on different graphs. The graphs used in this section include 10 different Waxman graphs, a Pure Random graph, a Locality graph, and a Transit-Stub graph. This section will help answer the question of whether the type of the graph used in the evaluation affects the heuristic's performance.

- *Evaluations on 100 different Waxman Graphs:* In Section 4.3 the heuristics are compared on 100 different 100 node Waxman graphs. The average node degrees of the graphs used in this section varies from a low of 3.04 to a high of 5.26. The average node degrees of all the 100 graphs used in the evaluation are listed in the Table 4.8. Comparing the heuristics performance averages on a large number of graphs helps the reader in assessing the heuristic's average performance with respect to its peers over a number of graphs.

- *Comparison to the Optimal Solution:* In Section 4.4 the heuristics' performance is compared to the optimal solution on a 10 node 16 edge graph. This helps us in understanding the heuristics' performance relative to the optimal value. Since finding the optimal solution is an NP-complete problem we use a small 16 edge graph in this evaluation.

Each edge in the graphs generated is labelled with its Euclidean length. In this evaluation the edge label represents the delay experienced on the edge (d_e). For an effective comparison of tree cost performance of the heuristics and to understand whether the relationship between the cost and delay associated with an edge affects the heuristics relative performance three different scenario's each assigning different cost values for the edges in the graph are considered:

- *EQUAL*: The cost of an edge has the same numerical value as the delay experienced on the edge. This scenario reflects the case when queuing delay is considered as the delay on the edge. When the queuing delay increases it is necessary to the increase the buffer size which increases the cost associated with the edge.

- *RANDOM*: The costs of the edges are a random permutation of the delays experienced on the edges in the graph. This scenario respresents the case where there is no relationship between the cost and delay associated with an edge, hence the cost associated with an edge is randomly assigned from the delays experienced on the graph edges.

- *REVERSE*: The costs of the edges are the delays in reverse order. That is the highest delay edge gets the lowest delay of any edge as its cost.

The heuristics are compared at various delay bound values at a session size of $m = 40$ (each multicast session services a set of 40 nodes). The delay bound is increased from 20 to 900 in increments of 20. The k value for DWH is taken from the following set $k \in \{20, 100, 160, 300, 900\}$ to capture a broad spectrum of DWH behavior.

4.1 Evaluation of the heuristics on 4 Waxman graphs

In this section the heuristics are evaluated on 4 different Waxman graphs with varying node degrees. In each figure, reporting the average tree cost values, each data point on the graph represents 100 multicast sessions used in the simulation. That is for every graph during the simulations, 100 different multicast destination sets are input to each heuristic at every delay bound value. The

average tree cost of the heuristic (which is plotted in the graphs) is the average of the tree costs for the 100 sessions in that configuration.

In each Figure the order of the heuristics in the legend also illustrates the differences in the relative performance of the heuristics. The heuristic exhibiting the worst relative performance is listed at the top of the legend while the heuristic exhibiting the best relative performance is listed at the bottom.

As all the heuristics merge the least-delay path (to a destination node) to the multicast tree, if the shortest paths computed based on edge-cost fail to meet the delay bound, the FCB value for all the heuristics is equal. Alternatively, unless there is no feasible path meeting the delay bound to any destination in the destination-set the heuristics will not block the request for the construction of the multicast tree. This has been confirmed through the evaluations conducted in this work. Hence the FCB results are not reported. Therefore only tree cost is used as a performance comparison metric henceforth.

Figure 4.1 compares the heuristics' tree cost performance for varying delay bound values when run on a 100 Node Waxman graph (Degree=6.02, $\alpha = 0.25$, $\beta = 0.21$) for the EQUAL cost scenario. The offline QDMR heuristic constructs multicast trees with lower cost than all other heuristics which is plausible as it has knowledge of all the destination nodes (which the online heuristics do not have) prior to the construction of the multicast tree and utilizes this information to lower the tree cost.

However it is interesting to note that the online destination node priority influenced variants of QDMR heuristic, DQDMR and DEPDT, construct multicast trees with the highest tree cost. This is because both DQDMR and DEPDT only assign variable priority to the destination nodes in the graph and not the more numerous tree nodes in the graph which would result in a higher reduction in the tree cost of the heuristics. Both TQDMR and TEPDT, the online tree node priority influenced variants of QDMR, are able to better utilize paths passing through the existing nodes in the multicast tree and hence construct lower cost multicast trees than DQDMR and DEPDT.

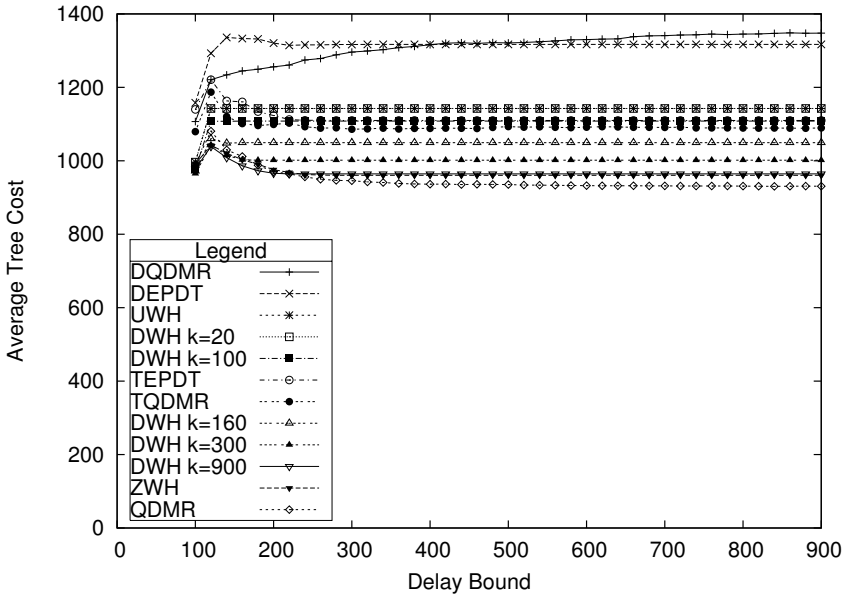

Figure 4.1: Tree cost versus Δ at $m = 40$ (EQUAL cost) Waxman graph: 100 Nodes, Degree=6.02, $\alpha = 0.25$, $\beta = 0.21$

The dynamic weight heuristic, DWH, at low k values (< 160) behaves like the least-cost path heuristic (UWH) and constructs multicast trees with tree costs similar to that of UWH. DWH at higher k values (≥ 160) constructs multicast trees with lower tree cost than even TQDMR and TEPDT (the best online variants of QDMR) and at high k values ($k = 900$) constructs multicast trees with tree cost similar to those constructed by ZWH (the least tree cost heuristic) and the offline QDMR.

This relative tree cost performance (with minor variations) is also seen in both the REVERSE and RANDOM cost scenarios as shown in Figure 4.2 and Figure 4.3. Here the DWH based heuristics have higher costs than the DQDMR based heuristics at lower delay bound values (≤ 200 in the RANDOM cost scenario and ≤ 300 in the REVERSE cost scenario) and lower costs at higher delay bound values.

DWH (at ≥ 300) and ZWH construct multicast trees with lower tree cost than TQDMR and TEPDT (the best online variants of QDMR) in both RANDOM and REVERSE scenarios. These

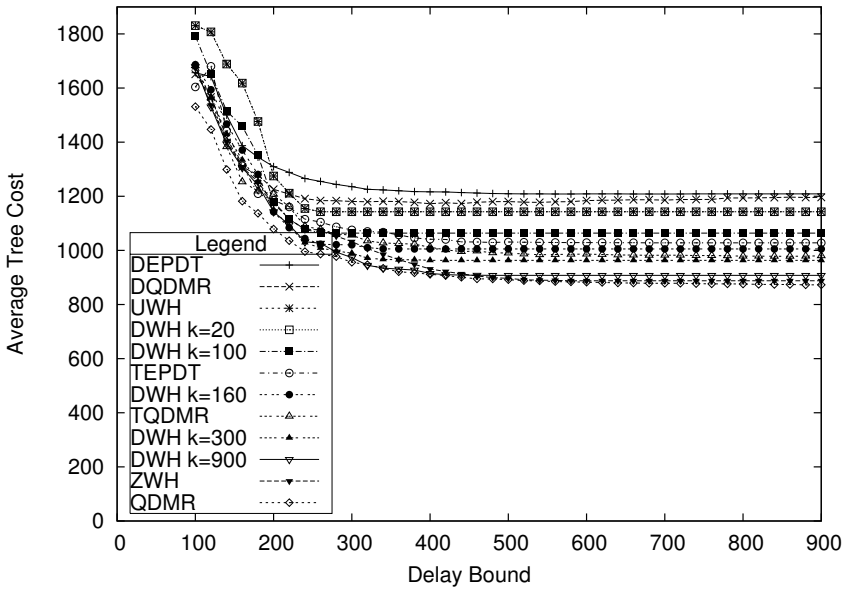

Figure 4.2: Tree cost versus Δ at $m = 40$ (RANDOM cost) Waxman graph: 100 Nodes, Degree=6.02, $\alpha = 0.25$, $\beta = 0.21$

results indicate that the edge-priority based heuristics (DWH, ZWH) have better tree cost performance over the node-priority based heuristics (DQDMR, DEPDT, TQDMR, TEPDT). Figures 4.4 - 4.12 illustrate similar differences for three other Waxman graphs with degrees (5.00, 4.04, and 3.04).

4.2 Evaluation on disparate graphs

As has been illustrated above, the performance of a multicast routing heuristic is dependent on the network/graph used for routing. To further illustrate the heuristics' relative performance, the heuristics are executed over several disparate graphs. In this evaluation phase ten different kinds of 100 node Waxman graphs were used. The properties of the Waxman graphs generated along with the parameters used during the generation process are shown in Table 4.1. Also used in the evaluation process are:

- *Pure Random* A 100 node Pure Random graph (generated with probability $p = 0.055$) with average node degree 5.36.

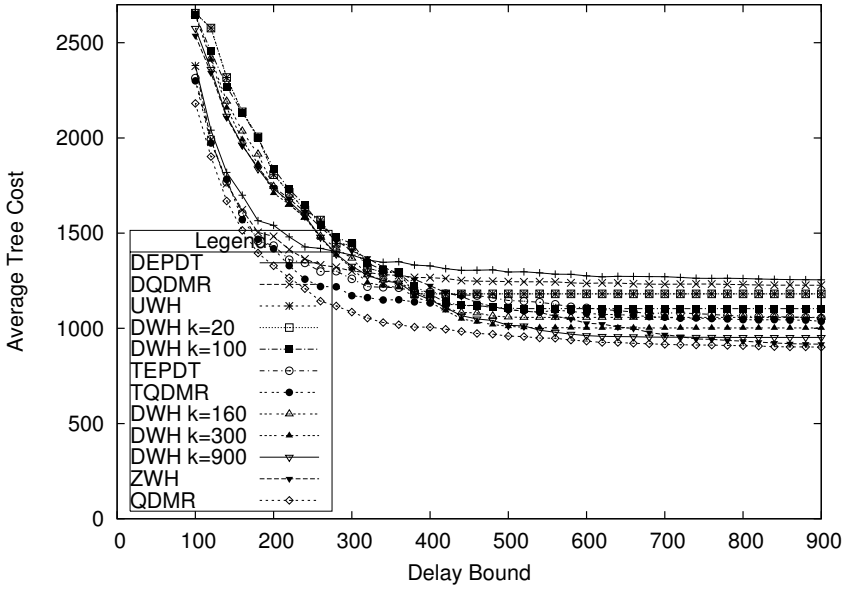

Figure 4.3: Tree cost versus Δ at $m = 40$ (REVERSE cost) Waxman graph: 100 Nodes, Degree=6.02, $\alpha = 0.25$, $\beta = 0.21$

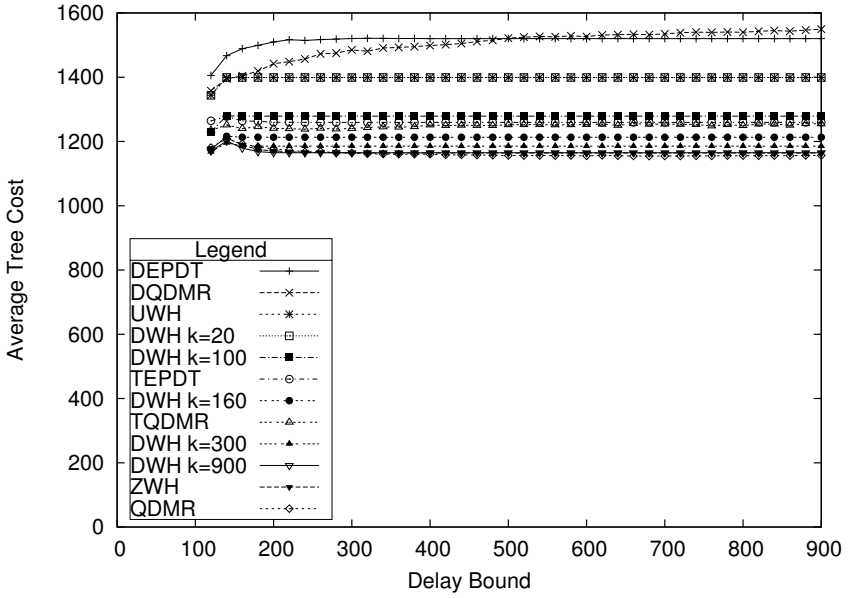

Figure 4.4: Tree cost versus Δ at $m = 40$ (EQUAL cost) Waxman graph: 100 Nodes, Degree=5.00, $\alpha = 0.235$, $\beta = 0.195$

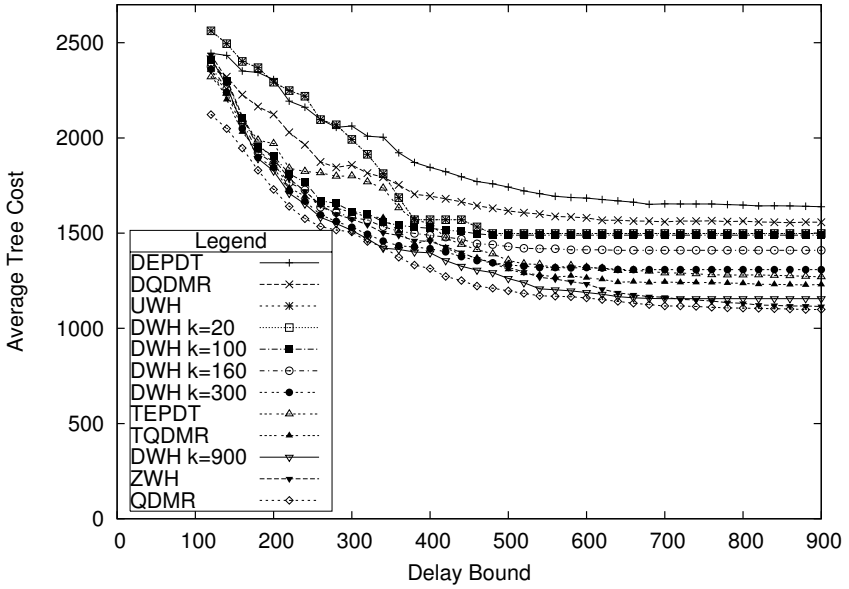

Figure 4.5: Tree cost versus Δ at $m = 40$ (REVERSE cost) Waxman graph: 100 Nodes, Degree=5.00, $\alpha = 0.235$, $\beta = 0.195$

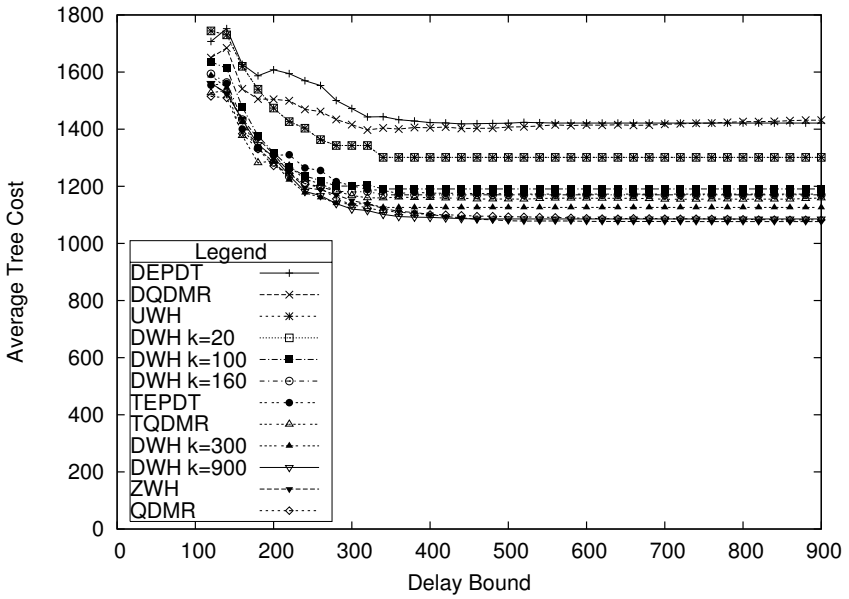

Figure 4.6: Tree cost versus Δ at $m = 40$ (RANDOM cost) Waxman graph: 100 Nodes, Degree=5.00, $\alpha = 0.235$, $\beta = 0.195$

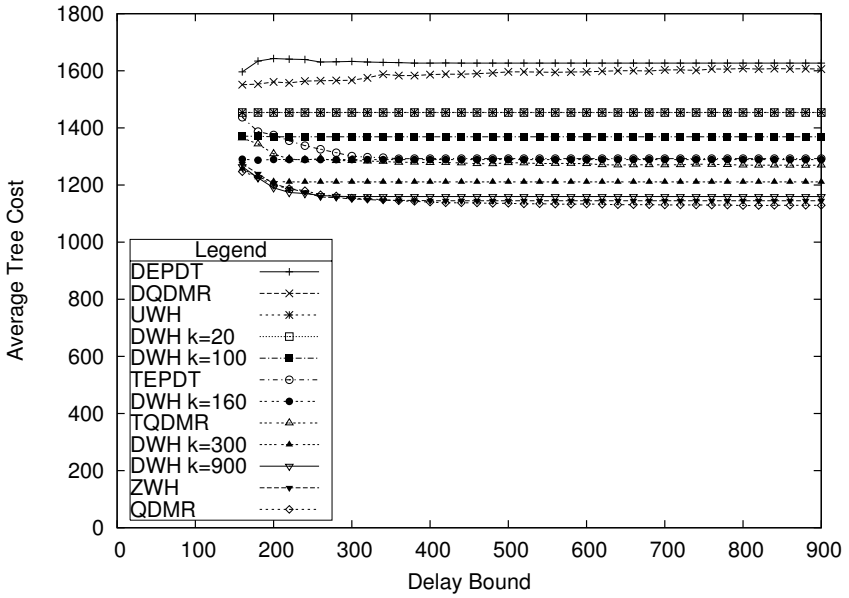

Figure 4.7: Tree cost versus Δ at $m = 40$ (EQUAL cost) Waxman graph: 100 Nodes, Degree=4.04, $\alpha = 0.210, \beta = 0.165$

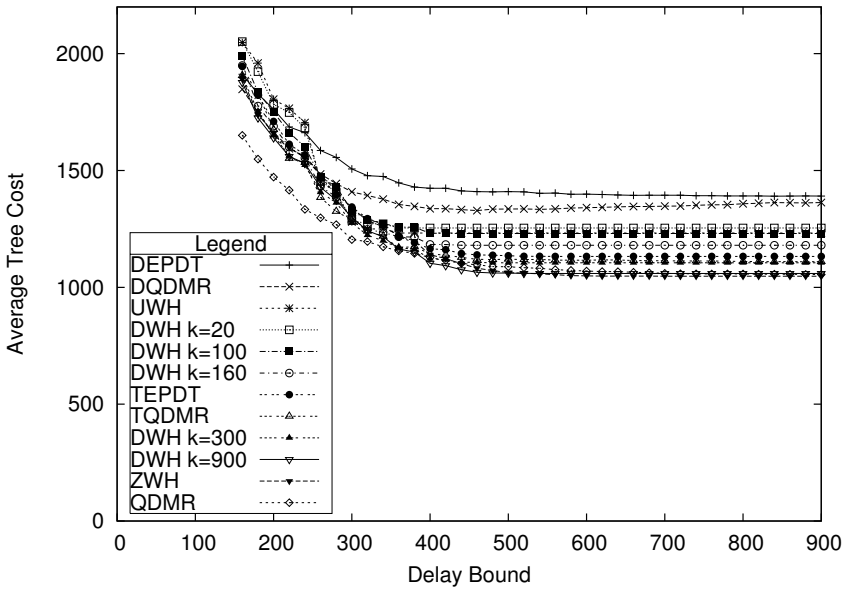

Figure 4.8: Tree cost versus Δ at $m = 40$ (REVERSE cost) Waxman graph: 100 Nodes, Degree=4.04, $\alpha = 0.210, \beta = 0.165$

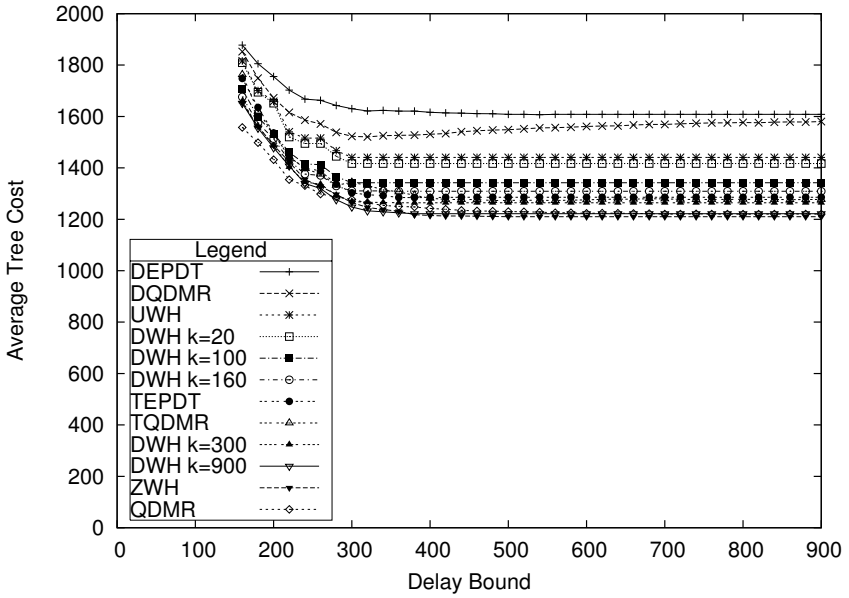

Figure 4.9: Tree cost versus Δ at $m = 40$ (RANDOM cost) Waxman graph: 100 Nodes, Degree=4.04, $\alpha = 0.210, \beta = 0.165$

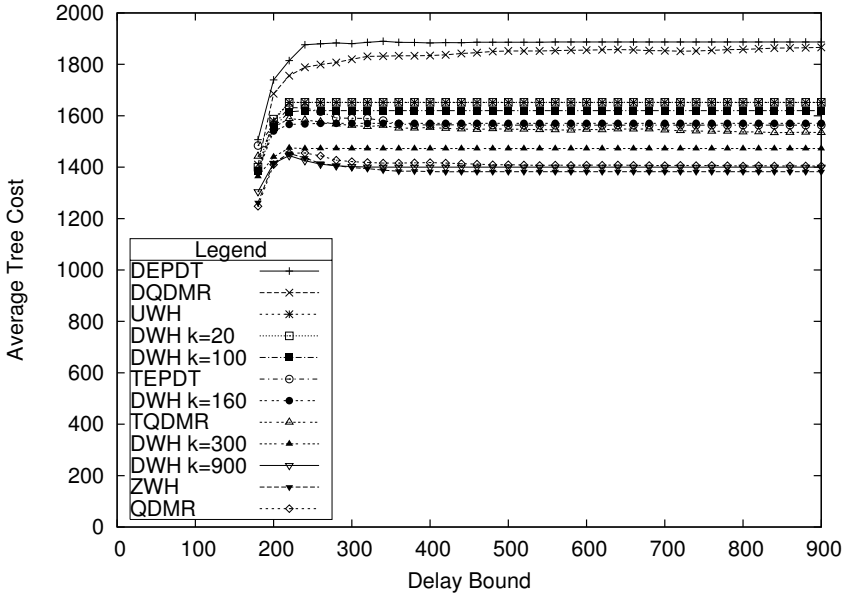

Figure 4.10: Tree cost versus Δ at $m = 40$ (EQUAL cost) Waxman graph: 100 Nodes, Degree=3.04, $\alpha = 0.190, \beta = 0.165$

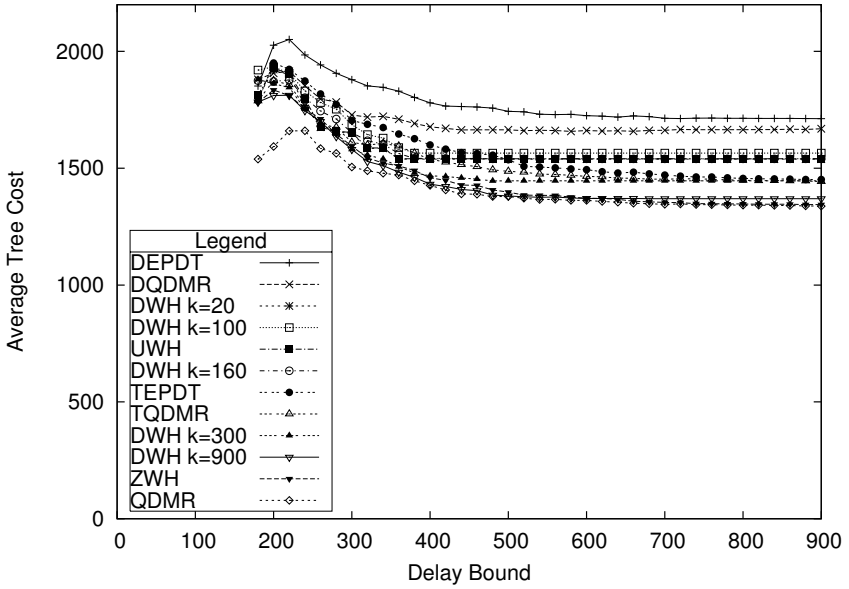

Figure 4.11: Tree cost versus Δ at $m = 40$ (REVERSE cost) Waxman graph: 100 Nodes, Degree=3.04, $\alpha = 0.190$, $\beta = 0.165$

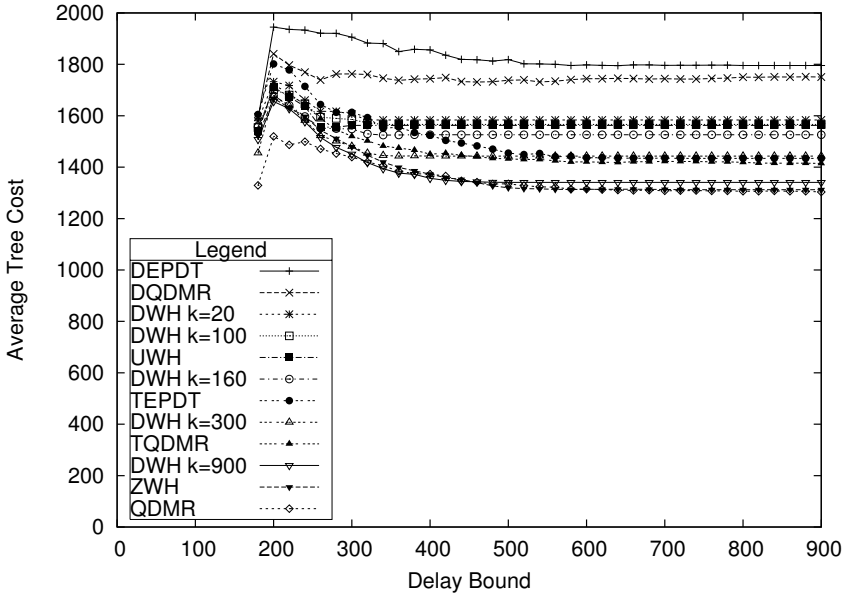

Figure 4.12: Tree cost versus Δ at $m = 40$ (RANDOM cost) Waxman graph: 100 Nodes, Degree=3.04, $\alpha = 0.190$, $\beta = 0.165$

Table 4.1: Ten Waxman Graphs

Name	Average Node Degree	Graph Parameters
Waxman1	3.46	$\alpha = 0.19, \beta = 0.15$
Waxman2	3.04	$\alpha = 0.19, \beta = 0.15$
Waxman3	3.36	$\alpha = 0.19, \beta = 0.15$
Waxman4	3.36	$\alpha = 0.19, \beta = 0.15$
Waxman5	3.30	$\alpha = 0.19, \beta = 0.15$
Waxman6	5.42	$\alpha = 0.24, \beta = 0.20$
Waxman7	5.40	$\alpha = 0.24, \beta = 0.20$
Waxman8	4.76	$\alpha = 0.24, \beta = 0.20$
Waxman9	5.30	$\alpha = 0.24, \beta = 0.20$
Waxman10	5.14	$\alpha = 0.24, \beta = 0.20$

- *Locality* A 100 node Locality graph (average node degree = 5.76), generated with the following parameters ($\alpha = 0.06, \beta = .005$, and $C = 35.35$).

- *Transit-Stub* A 100 node Transit-Stub graph with average node degree 3.74. The Transit-Stub graph has 1 transit domain of 4 nodes, 3 stub domains for each transit node and an average of 8 nodes per stub domain.

The performance of the heuristics over the ten Waxman graphs, pure random, locality, and transit-stub graphs (for the EQUAL cost scenario) is shown in Table 4.2. The evaluation results for the other two cost scenarios are shown in Table 4.4 (REVERSE cost) and Table 4.6 (RANDOM cost). In these tables the heuristics' performance order for a particular graph is listed in descending order along a row with the heuristic producing highest cost multicast trees listed at the left end and the heuristic producing lowest cost multicast trees listed at the right end.

Average cost of the multicast tree is the average over all the trees constructed for each graph. For example for each graph at each delay bound value 100 different 40 node multicast destination sets are fed to the heuristic and the heuristic attempts to construct a multicast tree for each set. If it is successful in its endeavor the multicast tree cost for that set is recorded. The delay bound

is varied from 20 to 900 in increments of 20 resulting in each heuristic attempting to construct a multicast tree for 4500 (100×45) instances. The average tree cost displayed in the table is the average over the successful among the 4500 instances. For example, for Waxman2 graph DEPDT constructs multicast trees with highest average tree cost of 1882.58 while DWH (at $k = 900$) constructs multicast trees with least average tree cost of 1388.71.

Further, to help in the relative performance assessment the heuristics are divided into three different groups as shown below.

1. *QDMR:* The only heuristic in this group is the offline QDMR.

2. *DQDMR:* The online variants of QDMR (DQDMR, DEPDT, TQDMR, TEPDT) are part of this group. These heuristics all assign variable priority to the nodes in the graph (either the destination node or the tree nodes) so as to construct low-cost multicast trees meeting the delay bound.

3. *DWH:* DWH and its variants (DWH, ZWH, UWH) comprise this group. These heuristics assign varying priorities to the in-tree edges in the graph.

Tables 4.3, 4.5, and 4.7 present an alternative representation of the relative tree cost performance of the different heuristics on these disparate graphs for the three different cost scenarios (EQUAL, REVERSE, and RANDOM respectively). Here the heuristics are all arranged in fixed columns so as to distinguish the performance of the heuristics in the different groups. The DQDMR group heuristics are listed followed by DWH and QDMR. This compact presentation helps the reader appreciate the average tree cost performance of each heuristic relative to its peers.

Figures 4.13 - 4.18 illustrate the tree cost performance of the three different groups of heuristics for EQUAL, REVERSE, and RANDOM cost scenarios. The figures plot the low and high values of the average tree cost observed in each heuristic group for each kind of graph. For example in the EQUAL cost scenario for Waxman1 graph DEPDT and TQDMR construct the highest and lowest

cost multicast trees in the DQDMR group and their tree cost values are plotted as the high and low tree cost values for DQDMR group respectively.

In order to improve the presentation clarity the performance of the 13 different graphs for each cost scenario is presented in two graphs. The first graph in each case presents the performance of the heuristics for graphs W1 (short for Waxman1), W2, W3, W4, W5, and W6 (Disparate Graphs-I). The second graph in each case presents the performance of the heuristics for graphs W7, W8, W9, W10, Random (Pure Random), Locality, and TS (Transit-Stub) (Disparate Graphs-II).

4.2.1 EQUAL cost scenario

From Table 4.2 it is clear that the graph used in the evaluation process affects the relative performance of the heuristics. This is because the heuristics' relative performance order is not consistent over any two graphs in the set. More specifically there is a variation in relative performance of the heuristics based on the type of the graph used in the evaluation process making it difficult to predict their relative performance in a new environment.

The general conclusions (relevant for most of the cases) that can be drawn from the data shown in the table are the following:

- The destination node priority influenced online variants of QDMR, (DQDMR and DEPDT) generally construct the highest cost multicast trees (with some exceptions Waxman7 and Waxman9).

- DWH (at $k = 900$), ZWH, and the offline QDMR consistently construct the lowest cost multicast trees.

- The tree node priority influenced online variants of QDMR, (TQDMR and TEPDT) generally construct multicast trees with lower tree cost than the destination node priority influenced online variants DQDMR and DEPDT.

- DWH (at $k = 900$) and ZWH consistently construct lower cost multicast trees than TQDMR

and TEPDT (the best online variants of the QDMR heuristic).

For the six Waxman graphs (W1 to W6) shown in Figure 4.13 it is interesting to note that the low value of the tree cost for DWH group heuristics is close to (sometimes lower than) the cost of the trees constructed by QDMR and is lower than the low value of the tree cost for the DQDMR group heuristics. Also the high value of the tree cost for DWH group heuristics is lower than the high value of the tree cost of the multicast trees constructed by the DQDMR group heuristics.

Similarly for the rest of the graphs (Disparate Graphs-II) shown in Figure 4.14 the low value of the tree cost for DWH group heuristics is close to the cost of the trees constructed by QDMR and is much lower than the low value of the tree cost for the DQDMR group heuristics. However the high value of the tree cost for the DWH group heuristics is close to (and for the Random case higher than) the high value of the multicast trees constructed by the DQDMR group heuristics.

4.2.2 REVERSE cost scenario

From the Table 4.4 (REVERSE cost scenario) it is again evident that the graph used in the evaluation process affects the relative performance of the heuristics. However, the general conclusions that can be drawn from the data shown in the table are the following :

- The destination node priority influenced online variants of QDMR, (DQDMR and DEPDT) and UWH generally construct the highest cost multicast trees.
- DWH (k = 900), TQDMR, ZWH, and the offline QDMR generally construct the lowest cost multicast trees.
- The tree node priority influenced online variants of QDMR, (TQDMR and TEPDT) generally construct multicast trees with lower tree cost than DQDMR and DEPDT.

From the disparate graphs listed in Figure 4.15 and Figure 4.16 it is clear that the low value of the tree cost for DWH group heuristics is lower than or close to the low value of the tree cost for the DQDMR group heuristics and the high value of tree cost for the DWH group heuristics is

Table 4.2: Heuristics' Average Tree Cost Performance Order Over Different Graphs (EQUAL cost)

	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	TQDMR	DWH-160	DWH-300	DWH-900	ZWH	QDMR
Waxman1	Cost	1606.39	1597.88	1499.60	1498.73	1439.59	1379.41	1341.30	1313.49	1249.75	1210.84	1204.70	1200.42
	H	DEPDT	DQDMR	DWH-020	UWH	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	QDMR	DWH-900	ZWH
Waxman2	Cost	1882.58	1841.61	1651.72	1651.36	1619.28	1571.83	1570.04	1549.75	1472.51	1413.49	1402.63	1388.71
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	DWH-160	TEPDT	TQDMR	DWH-300	DWH-900	QDMR	ZWH
Waxman3	Cost	1762.70	1715.14	1638.44	1625.39	1529.41	1454.08	1451.92	1425.60	1368.40	1306.15	1300.04	1293.98
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	TQDMR	DWH-160	DWH-300	DWH-900	ZWH	QDMR
Waxman4	Cost	1727.43	1714.13	1641.33	1587.02	1535.66	1499.75	1477.12	1475.53	1415.05	1383.05	1381.21	1380.94
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	DWH-160	DWH-300	TEPDT	TQDMR	DWH-900	QDMR	ZWH
Waxman5	Cost	1881.31	1866.40	1651.40	1651.40	1651.40	1644.18	1564.90	1513.29	1491.35	1410.73	1396.35	1383.17
	H	DEPDT	DQDMR	UWH	DWH-020	TEPDT	DWH-100	TQDMR	DWH-160	DWH-300	DWH-900	ZWH	QDMR
Waxman6	Cost	1338.45	1328.80	1155.50	1155.50	1133.27	1124.44	1116.84	1068.73	1030.96	1002.82	999.26	980.36
	H	DWH-020	UWH	DEPDT	DQDMR	TEPDT	DWH-100	TQDMR	DWH-160	DWH-300	DWH-900	ZWH	QDMR
Waxman7	Cost	1419.73	1413.20	1355.67	1354.20	1193.13	1169.93	1153.26	1106.86	1055.71	1028.50	1020.52	978.37
	H	DEPDT	DQDMR	DWH-020	UWH	DWH-100	TEPDT	TQDMR	DWH-160	DWH-300	DWH-900	ZWH	QDMR
Waxman8	Cost	1459.79	1441.24	1418.62	1389.90	1264.90	1238.27	1202.08	1167.65	1099.06	1063.15	1062.34	1059.08
	H	DWH-020	UWH	DQDMR	DEPDT	DWH-100	TEPDT	TQDMR	DWH-160	DWH-300	DWH-900	ZWH	QDMR
Waxman9	Cost	1438.71	1418.32	1389.38	1386.90	1224.70	1191.48	1163.29	1138.58	1085.83	1042.04	1036.78	1014.21
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	TQDMR	DWH-160	DWH-300	DWH-900	ZWH	QDMR
Waxman10	Cost	1492.63	1483.94	1337.76	1337.76	1278.61	1275.88	1265.43	1219.48	1188.14	1165.32	1164.90	1153.77
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	DWH-900	ZWH	QDMR
Random	Cost	2221.06	2190.27	2058.47	2058.47	1970.66	1932.04	1923.36	1867.23	1775.37	1699.78	1681.97	1642.61
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	DWH-900	ZWH	QDMR
Locality	Cost	2156.75	2131.82	1990.53	1990.53	1897.74	1861.64	1844.57	1812.09	1730.47	1661.74	1653.21	1623.47
	H	DQDMR	DEPDT	UWH	DWH-020	DWH-100	TEPDT	TQDMR	DWH-160	QDMR	DWH-300	DWH-900	ZWH
TS	Cost	973.28	973.16	970.28	970.17	929.60	910.78	907.15	904.50	891.79	890.18	883.51	883.07

Table 4.3: Heuristics' Average Tree Cost Performance Over Different Graphs (EQUAL cost)

	DEPDT	DQDMR	TEPDT	TQDMR	UWH	DWH-020	DWH-100	DWH-160	DWH-300	ZWH	DWH-900	QDMR
Waxman1	1606.39	1597.88	1379.41	1341.30	1499.60	1498.73	1439.59	1313.49	1249.75	1204.70	1210.84	1200.42
Waxman2	1882.58	1841.61	1571.83	1549.75	1651.36	1651.72	1619.28	1570.04	1472.51	1388.71	1402.63	1413.49
Waxman3	1762.70	1715.14	1451.92	1425.60	1638.44	1625.39	1529.41	1454.08	1368.40	1293.98	1306.15	1300.04
Waxman4	1727.43	1714.13	1499.75	1477.12	1641.33	1587.02	1535.66	1475.53	1415.05	1381.21	1383.05	1380.94
Waxman5	1881.31	1866.40	1513.29	1491.35	1651.40	1651.40	1651.40	1644.18	1564.90	1383.17	1410.73	1396.35
Waxman6	1338.45	1328.80	1133.27	1116.84	1155.50	1155.50	1124.44	1068.73	1030.96	999.26	1002.82	980.36
Waxman7	1355.67	1354.20	1193.13	1153.26	1413.20	1419.73	1169.93	1106.86	1055.71	1020.52	1028.50	978.37
Waxman8	1459.79	1441.24	1238.27	1202.08	1389.90	1418.62	1264.90	1167.65	1099.06	1062.34	1063.15	1059.08
Waxman9	1386.90	1389.38	1191.48	1163.29	1418.32	1438.71	1224.70	1138.58	1085.83	1036.78	1042.04	1014.21
Waxman10	1492.63	1483.94	1275.88	1265.43	1337.76	1337.76	1278.61	1219.48	1188.14	1164.90	1165.32	1153.77
Random	2221.06	2190.27	1932.04	1867.23	2058.47	2058.47	1970.66	1923.36	1775.37	1681.97	1699.78	1642.61
Locality	2156.75	2131.82	1861.64	1812.09	1990.53	1990.53	1897.74	1844.57	1730.47	1653.21	1661.74	1623.47
TS	973.16	973.28	910.78	907.15	970.28	970.17	929.60	904.50	890.18	883.07	883.51	891.79

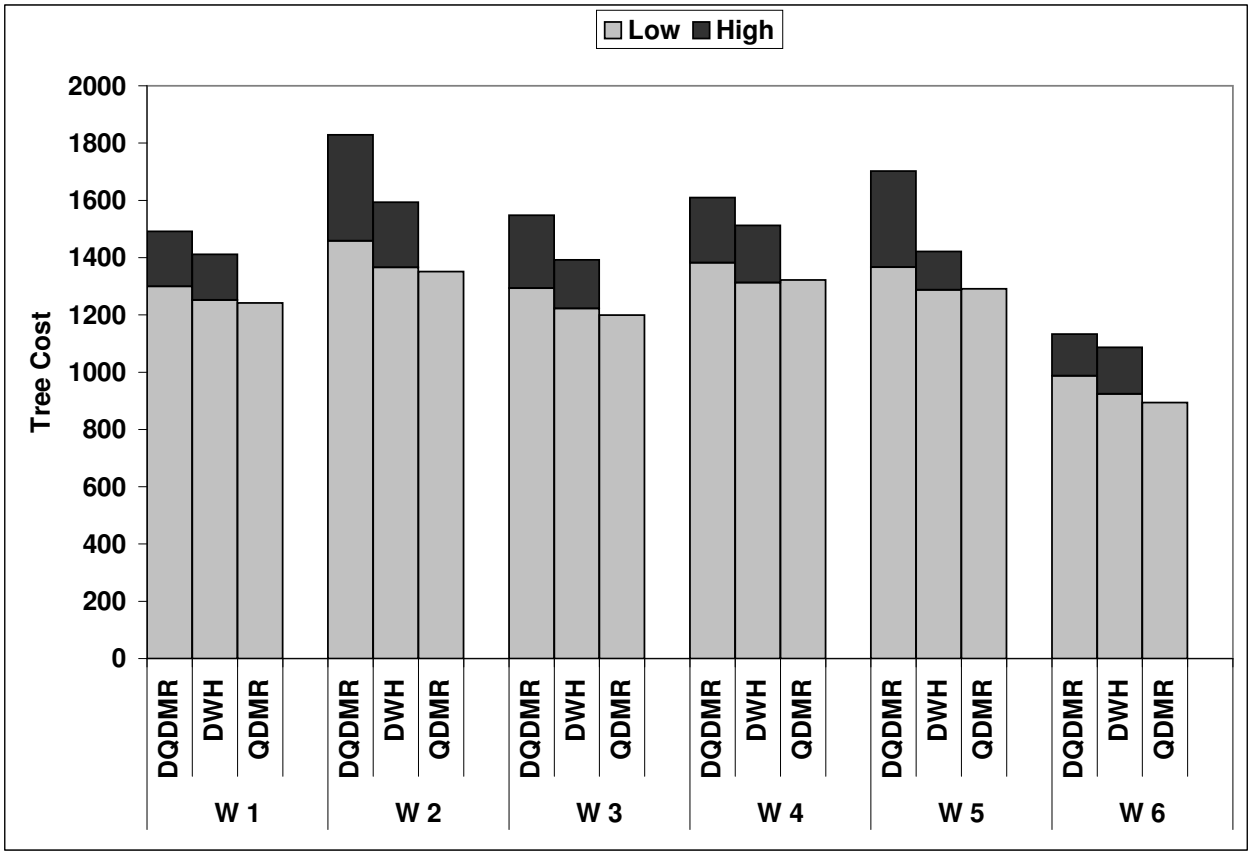

Figure 4.13: Heuristics Performance on Disparate Graphs-I (EQUAL cost)

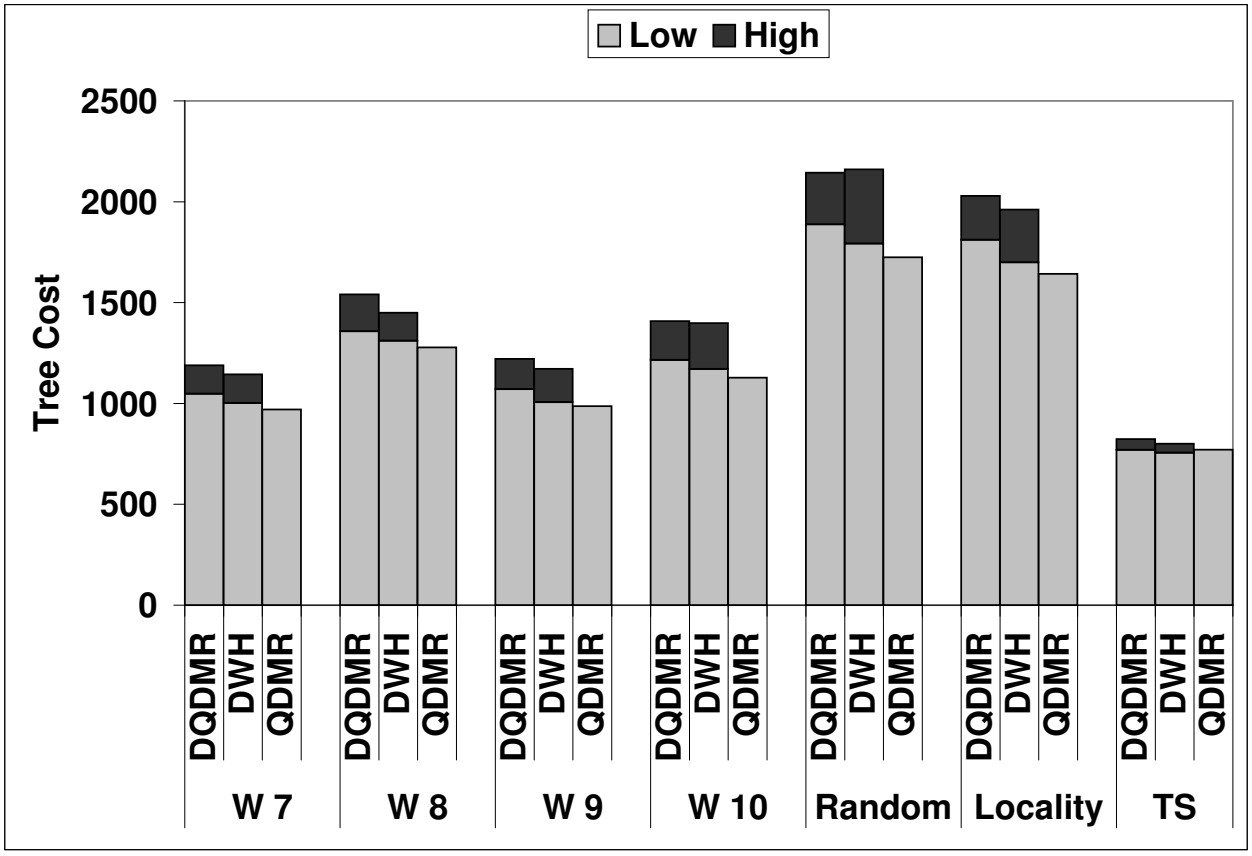

Figure 4.14: Heuristics Performance on Disparate Graphs-II (EQUAL cost)

lower than the high value of the tree cost of the multicast trees constructed by the DQDMR group heuristics in 9 of the 13 graphs used in this section.

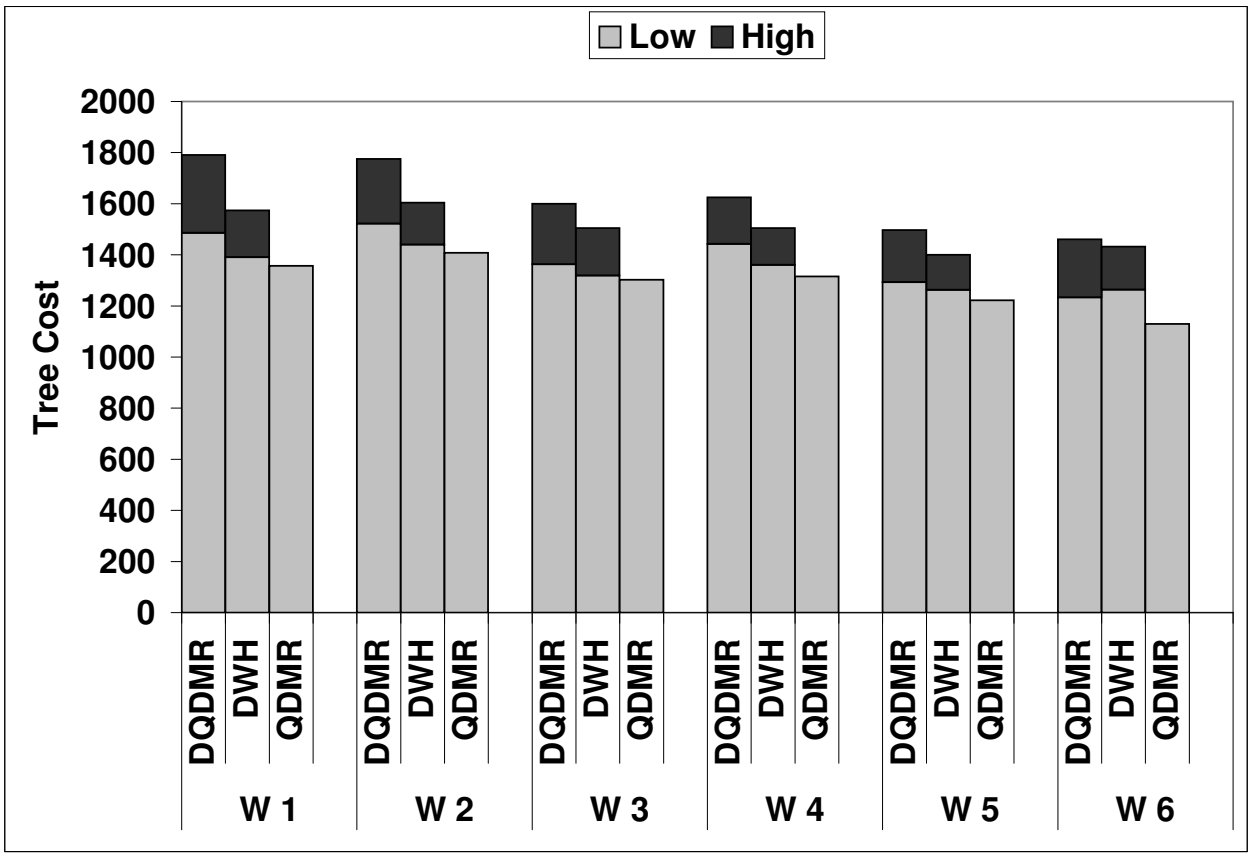

Figure 4.15: Heuristics Performance on Disparate Graphs-I (REVERSE cost)

4.2.3 RANDOM cost scenario

As explained above the graph used in the evaluation process affects the relative performance of the heuristics as is evidenced in Table 4.6 (RANDOM cost scenario). However, the following general conclusions can be drawn from the data shown in the table:

- DWH (k = 900), ZWH, and the offline QDMR consistently construct the lowest cost multicast trees.

Table 4.4: Heuristics' Average Tree Cost Performance Order Over Different Graphs (REVERSE cost)

	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	DWH-900	ZWH	QDMR
Waxman1	Cost	1791.09	1717.04	1573.56	1563.14	1561.11	1533.71	1520.70	1485.50	1464.98	1395.33	1390.65	1357.18
	H	DEPDT	DQDMR	DWH-100	DWH-160	UWH	DWH-020	TEPDT	TQDMR	DWH-300	ZWH	DWH-900	QDMR
Waxman2	Cost	1774.87	1692.54	1604.22	1578.16	1574.09	1571.62	1562.82	1521.64	1495.71	1440.82	1440.13	1408.03
	H	DEPDT	DQDMR	DWH-100	UWH	DWH-160	DWH-020	DWH-300	TEPDT	TQDMR	ZWH	DWH-900	QDMR
Waxman3	Cost	1599.62	1528.91	1504.40	1490.06	1475.15	1469.30	1390.49	1383.22	1362.66	1324.25	1319.36	1302.59
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	DWH-900	ZWH	QDMR
Waxman4	Cost	1625.03	1571.02	1504.88	1499.39	1484.65	1481.05	1463.86	1442.39	1426.61	1363.97	1360.40	1315.66
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	DWH-160	DWH-300	TEPDT	TQDMR	DWH-900	ZWH	QDMR
Waxman5	Cost	1497.35	1411.04	1400.29	1400.29	1400.29	1400.29	1358.44	1331.66	1292.69	1269.71	1262.40	1221.91
	H	DEPDT	UWH	DWH-020	DQDMR	DWH-100	DWH-160	DWH-300	ZWH	TEPDT	DWH-900	TQDMR	QDMR
Waxman6	Cost	1460.87	1432.09	1432.09	1411.26	1375.18	1322.47	1288.66	1286.61	1272.59	1264.04	1233.13	1129.32
	H	UWH	DWH-020	DWH-100	DEPDT	DQDMR	DWH-160	ZWH	DWH-300	DWH-900	TEPDT	TQDMR	QDMR
Waxman7	Cost	1357.19	1340.86	1290.67	1288.50	1269.00	1253.60	1221.11	1215.50	1193.83	1178.41	1164.71	1099.38
	H	UWH	DWH-020	DEPDT	DWH-100	DQDMR	DWH-160	TEPDT	ZWH	DWH-300	TQDMR	DWH-900	QDMR
Waxman8	Cost	1652.92	1573.90	1567.44	1502.29	1467.37	1457.13	1444.97	1418.04	1409.78	1373.92	1371.79	1258.47
	H	UWH	DWH-020	DEPDT	DWH-100	DQDMR	DWH-160	TEPDT	ZWH	TQDMR	DWH-300	DWH-900	QDMR
Waxman9	Cost	1324.68	1311.46	1197.89	1180.37	1176.23	1141.89	1126.86	1119.41	1110.78	1107.69	1088.77	1008.58
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	DWH-160	TEPDT	DWH-300	ZWH	TQDMR	DWH-900	QDMR
Waxman10	Cost	1721.52	1636.56	1626.48	1626.48	1494.59	1457.57	1431.40	1403.10	1380.83	1376.88	1353.08	1262.36
	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	DWH-160	ZWH	DWH-300	DWH-900	TQDMR	QDMR
Random	Cost	2452.77	2322.28	2228.61	2228.61	2195.94	2163.93	2153.64	2124.01	2118.71	2067.48	2049.50	1887.16
	H	DEPDT	DQDMR	UWH	DWH-020	TEPDT	DWH-100	DWH-160	ZWH	DWH-300	DWH-900	TQDMR	QDMR
Locality	Cost	2462.25	2328.27	2224.27	2224.27	2174.28	2163.74	2132.70	2122.56	2095.87	2066.06	2054.96	1889.02
	H	UWH	DWH-020	DQDMR	DEPDT	DWH-100	TEPDT	TQDMR	DWH-160	DWH-300	DWH-900	ZWH	QDMR
TS	Cost	839.12	839.12	832.31	828.18	816.85	814.57	808.51	804.40	791.15	784.57	783.84	777.10

Table 4.5: Heuristics' Average Tree Cost Performance Over Different Graphs (REVERSE cost)

	DEPDT	DQDMR	TEPDT	TQDMR	UWH	DWH-020	DWH-100	DWH-160	DWH-300	ZWH	DWH-900	QDMR
Waxman1	1791.09	1717.04	1533.71	1485.50	1573.56	1563.14	1561.11	1520.70	1464.98	1390.65	1395.33	1357.18
Waxman2	1774.87	1692.54	1562.82	1521.64	1574.09	1571.62	1604.22	1578.16	1495.71	1440.82	1440.13	1408.03
Waxman3	1599.62	1528.91	1383.22	1362.66	1490.06	1469.30	1504.40	1475.15	1390.49	1324.25	1319.36	1302.59
Waxman4	1625.03	1571.02	1481.05	1442.39	1504.88	1499.39	1484.65	1463.86	1426.61	1360.40	1363.97	1315.66
Waxman5	1497.35	1411.04	1331.66	1292.69	1400.29	1400.29	1400.29	1400.29	1358.44	1262.40	1269.71	1221.91
Waxman6	1460.87	1411.26	1272.59	1233.13	1432.09	1432.09	1375.18	1322.47	1288.66	1286.61	1264.04	1129.32
Waxman7	1288.50	1269.00	1178.41	1164.71	1357.19	1340.86	1290.67	1253.60	1215.50	1221.11	1193.83	1099.38
Waxman8	1567.44	1467.37	1444.97	1373.92	1652.92	1573.90	1502.29	1457.13	1409.78	1418.04	1371.79	1258.47
Waxman9	1197.89	1176.23	1126.86	1110.78	1324.68	1311.46	1180.37	1141.89	1107.69	1119.41	1088.77	1008.58
Waxman10	1721.52	1636.56	1431.40	1376.88	1626.48	1626.48	1494.59	1457.57	1403.10	1380.83	1353.08	1262.36
Random	2452.77	2322.28	2163.93	2049.50	2228.61	2228.61	2195.94	2153.64	2118.71	2124.01	2067.48	1887.16
Locality	2462.25	2328.27	2174.28	2054.96	2224.27	2224.27	2163.74	2132.70	2095.87	2122.56	2066.06	1889.02
TS	828.18	832.31	814.57	808.51	839.12	839.12	816.85	804.40	791.15	783.84	784.57	777.10

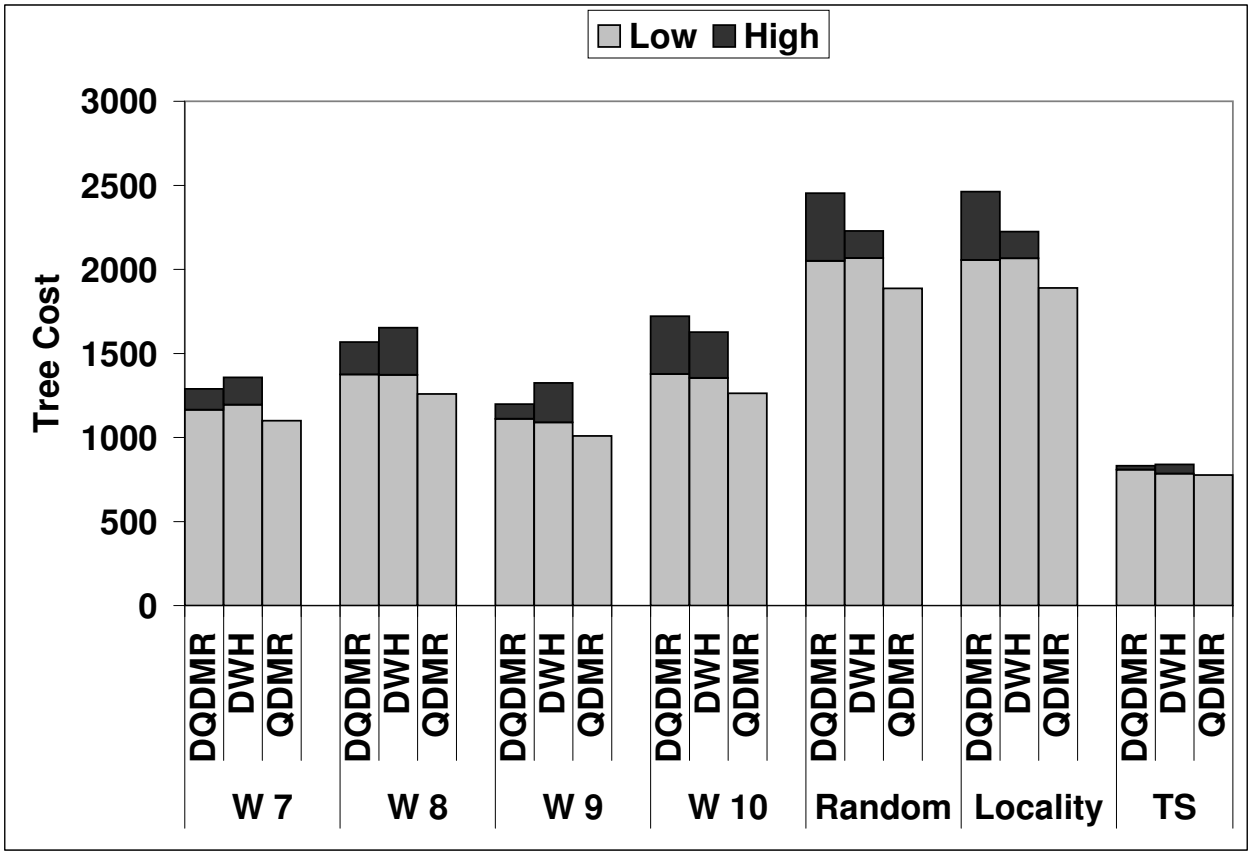

Figure 4.16: Heuristics Performance on Disparate Graphs-II (REVERSE cost)

- The destination node priority influenced variants of QDMR (DQDMR and DEPDT) generally construct the highest cost multicast trees.

- The tree node priority influenced online variants of QDMR, (TQDMR and TEPDT) consistently construct lower cost multicast trees than the destination node priority influenced online variants DQDMR and DEPDT.

- DWH with increasing k values constructs lower cost multicast trees.

From the disparate graphs listed in Figure 4.17 and Figure 4.18 it is evident that the low value of the tree cost for DWH group heuristics is close to (and sometimes lower than) the cost of the trees constructed by QDMR and much lower than the low value of the tree cost for the DQDMR group heuristics. Here, the high value of tree cost for the DWH group heuristics is mostly (except for the Random graph case) lower than the high value of the tree cost of the multicast trees constructed by the DQDMR group heuristics for all the graphs. These results illustrate that the DWH based heuristics construct lower cost multicast trees more consistently than the DQDMR based heuristics which is further confirmed through evaluations of the heuristics over 100 graphs in Section 4.3.

4.3 Evaluations on 100 different Waxman Graphs:

If a heuristic's relative performance is deemed to be dependent on the graph used in the evaluation process then it is imperative for the heuristic designer to test the heuristic on the target graphs or to extensively evaluate the heuristic's relative performance on graphs resembling target networks/graphs.

In this section the heuristics are all compared on one hundred different 100 node Waxman graphs. The properties of the Waxman graphs generated along with the parameters used during the generation process are shown in Table 4.8. The heuristic runs are simulated on these 100 different Waxman graphs for an extensive comparison of the heuristics' performance. All the heuristics are compared against DQDMR to measure their relative performance.

Table 4.6: Heuristics' Average Tree Cost Performance Order Over Different Graphs (RANDOM cost)

Waxman1	H	DEPDT	DQDMR	DWH-100	UWH	DWH-160	DWH-020	TEPDT	TQDMR	DWH-300	ZWH	DWH-900	QDMR
	Cost	1491.54	1454.90	1411.23	1383.48	1359.24	1347.35	1322.25	1299.98	1295.52	1254.49	1252.03	1241.84
Waxman2	H	DEPDT	DQDMR	DWH-020	DWH-100	UWH	DWH-160	TEPDT	DWH-300	TQDMR	DWH-900	ZWH	QDMR
	Cost	1829.01	1747.39	1593.21	1574.77	1568.33	1534.46	1495.52	1459.51	1458.28	1376.12	1366.37	1351.28
Waxman3	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	DWH-160	TEPDT	TQDMR	DWH-300	DWH-900	ZWH	QDMR
	Cost	1547.74	1499.73	1392.38	1376.72	1344.25	1338.90	1312.62	1293.30	1256.45	1224.30	1222.47	1199.50
Waxman4	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	QDMR	DWH-900	ZWH
	Cost	1609.50	1596.06	1512.28	1486.61	1436.76	1395.97	1393.17	1381.84	1352.32	1321.98	1314.33	1312.52
Waxman5	H	DEPDT	DQDMR	DWH-160	DWH-100	UWH	DWH-020	DWH-300	TEPDT	TQDMR	DWH-900	QDMR	ZWH
	Cost	1702.43	1638.43	1421.18	1405.06	1404.96	1404.96	1400.87	1395.06	1366.55	1303.83	1291.29	1287.67
Waxman6	H	DEPDT	DQDMR	UWH	DWH-020	TEPDT	DWH-100	TQDMR	DWH-160	DWH-300	ZWH	DWH-900	QDMR
	Cost	1132.76	1112.51	1086.78	1086.78	1022.06	1012.00	987.65	981.92	950.99	927.31	924.21	894.04
Waxman7	H	DQDMR	DEPDT	UWH	DWH-020	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	ZWH	DWH-900	QDMR
	Cost	1189.38	1189.07	1144.06	1116.22	1084.75	1061.05	1055.76	1047.52	1020.69	1005.18	1001.47	970.34
Waxman8	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	DWH-160	TEPDT	TQDMR	DWH-300	ZWH	DWH-900	QDMR
	Cost	1540.42	1515.97	1450.09	1442.12	1425.92	1381.76	1374.01	1357.51	1337.73	1313.08	1310.65	1277.63
Waxman9	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	ZWH	DWH-900	QDMR
	Cost	1221.03	1216.86	1172.23	1170.49	1137.16	1094.59	1084.49	1071.39	1030.57	1007.91	1006.36	986.91
Waxman10	H	DEPDT	UWH	DWH-020	DQDMR	DWH-100	DWH-160	TEPDT	DWH-300	TQDMR	ZWH	DWH-900	QDMR
	Cost	1407.80	1398.75	1398.75	1372.85	1313.37	1264.51	1230.18	1217.88	1214.99	1176.48	1170.56	1128.42
Random	H	UWH	DWH-020	DEPDT	DQDMR	DWH-100	DWH-160	TEPDT	TQDMR	DWH-300	ZWH	DWH-900	QDMR
	Cost	2160.00	2160.00	2144.12	2114.32	2085.35	1989.90	1917.06	1887.61	1871.86	1793.99	1792.94	1724.80
Locality	H	DEPDT	DQDMR	UWH	DWH-020	DWH-100	TEPDT	DWH-160	TQDMR	DWH-300	DWH-900	ZWH	QDMR
	Cost	2029.41	2003.79	1961.38	1961.38	1888.68	1828.77	1817.28	1811.43	1742.88	1704.84	1699.15	1643.11
TS	H	DQDMR	DEPDT	UWH	DWH-020	DWH-100	TEPDT	QDMR	DWH-160	TQDMR	DWH-300	ZWH	DWH-900
	Cost	823.66	818.56	800.79	800.79	779.96	777.03	771.32	770.95	769.97	758.66	755.78	755.54

Table 4.7: Heuristics' Average Tree Cost Performance Over Different Graphs (RANDOM cost)

	DEPDT	DQDMR	TEPDT	TQDMR	UWH	DWH-020	DWH-100	DWH-160	DWH-300	ZWH	DWH-900	QDMR
Waxman1	1491.54	1454.90	1322.25	1299.98	1383.48	1347.35	1411.23	1359.24	1295.52	1254.49	1252.03	1241.84
Waxman2	1829.01	1747.39	1495.52	1458.28	1568.33	1593.21	1574.77	1534.46	1459.51	1366.37	1376.12	1351.28
Waxman3	1547.74	1499.73	1312.62	1293.30	1392.38	1376.72	1344.25	1338.90	1256.45	1222.47	1224.30	1199.50
Waxman4	1609.50	1596.06	1395.97	1381.84	1512.28	1486.61	1436.76	1393.17	1352.32	1312.52	1314.33	1321.98
Waxman5	1702.43	1638.43	1395.06	1366.55	1404.96	1404.96	1405.06	1421.18	1400.87	1287.67	1303.83	1291.29
Waxman6	1132.76	1112.51	1022.06	987.65	1086.78	1086.78	1012.00	981.92	950.99	927.31	924.21	894.04
Waxman7	1189.07	1189.38	1061.05	1047.52	1144.06	1116.22	1084.75	1055.76	1020.69	1005.18	1001.47	970.34
Waxman8	1540.42	1515.97	1374.01	1357.51	1450.09	1442.12	1425.92	1381.76	1337.73	1313.08	1310.65	1277.63
Waxman9	1221.03	1216.86	1094.59	1071.39	1172.23	1170.49	1137.16	1084.49	1030.57	1007.91	1006.36	986.91
Waxman10	1407.80	1372.85	1230.18	1214.99	1398.75	1398.75	1313.37	1264.51	1217.88	1176.48	1170.56	1128.42
Random	2144.12	2114.32	1917.06	1887.61	2160.00	2160.00	2085.35	1989.90	1871.86	1793.99	1792.94	1724.80
Locality	2029.41	2003.79	1828.77	1811.43	1961.38	1961.38	1888.68	1817.28	1742.88	1699.15	1704.84	1643.11
TS	818.56	823.66	777.03	769.97	800.79	800.79	779.96	770.95	758.66	755.78	755.54	771.32

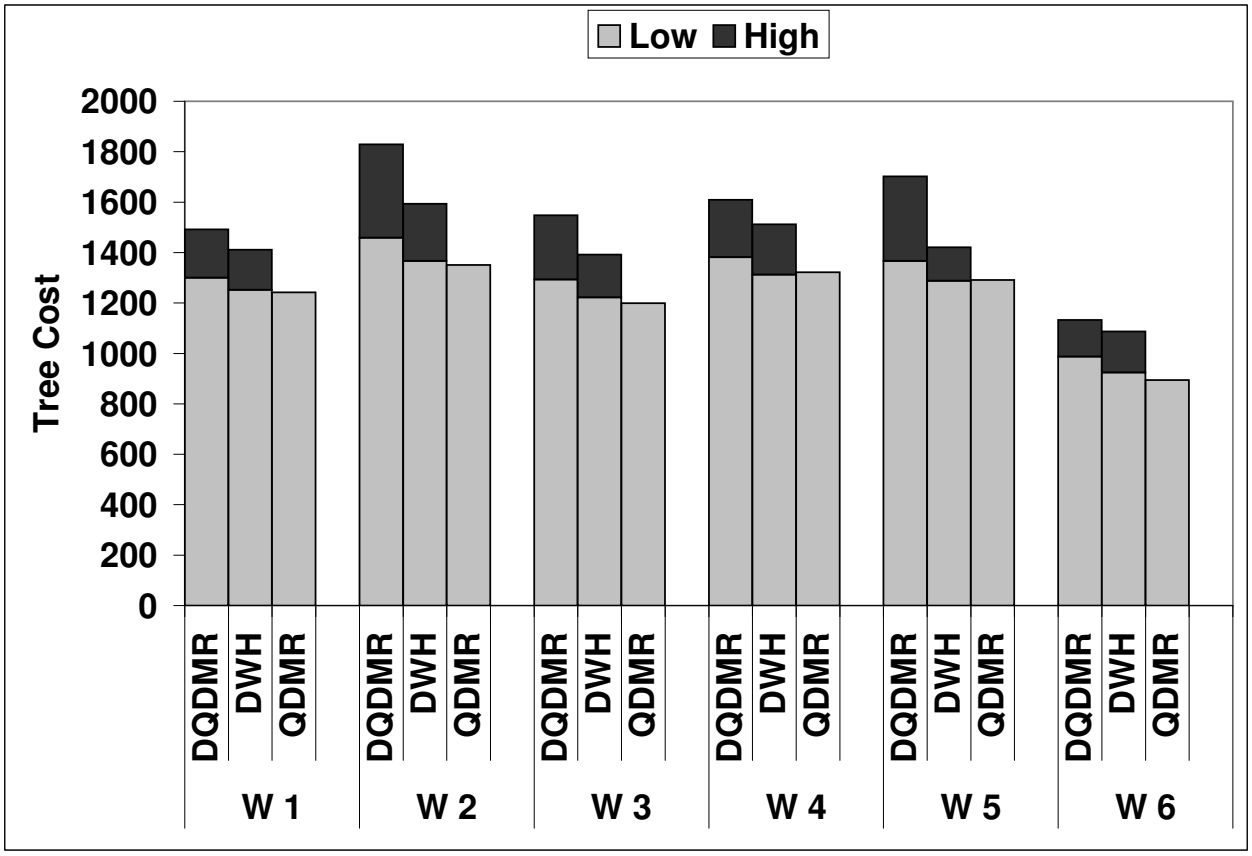

Figure 4.17: Heuristics Performance on Disparate Graphs-I (RANDOM cost)

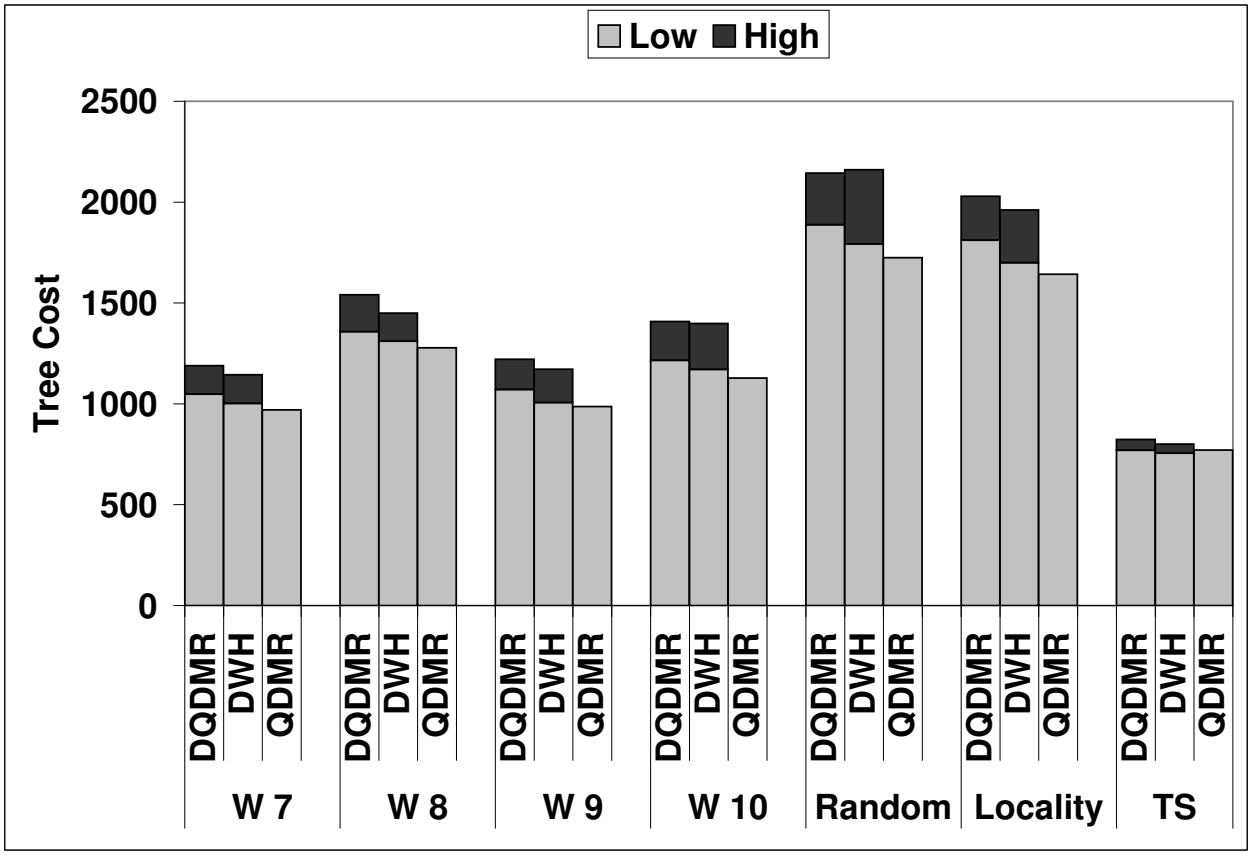

Figure 4.18: Heuristics Performance on Disparate Graphs-II (RANDOM cost)

Table 4.8: Properties of the 100 Waxman Graphs

Waxman	Average Node Degree	Graph Parameters
01 - 05	3.46, 3.04, 3.36, 3.36, 3.30	$\alpha = 0.190, \beta = 0.150$
05 - 10	3.32, 3.76, 3.24, 3.34, 3.30	$\alpha = 0.195, \beta = 0.150$
10 - 15	3.50, 3.84, 3.30, 3.40, 3.50	$\alpha = 0.195, \beta = 0.155$
15 - 20	3.56, 3.92, 3.36, 3.78, 3.44	$\alpha = 0.200, \beta = 0.155$
20 - 25	3.74, 4.16, 3.50, 3.88, 4.22	$\alpha = 0.200, \beta = 0.160$
25 - 30	3.86, 4.20, 3.66, 3.56, 3.94	$\alpha = 0.205, \beta = 0.160$
30 - 35	3.72, 3.94, 4.30, 3.88, 3.68	$\alpha = 0.205, \beta = 0.165$
35 - 40	3.86, 4.00, 4.04, 4.34, 4.08	$\alpha = 0.210, \beta = 0.165$
40 - 45	3.96, 4.26, 4.20, 4.46, 4.34	$\alpha = 0.210, \beta = 0.170$
45 - 50	4.04, 4.38, 4.30, 4.56, 4.30	$\alpha = 0.215, \beta = 0.170$
50 - 55	4.24, 4.48, 3.86, 4.44, 4.68	$\alpha = 0.215, \beta = 0.175$
55 - 60	4.30, 4.48, 3.94, 3.88, 4.58	$\alpha = 0.220, \beta = 0.175$
60 - 65	4.44, 4.58, 4.14, 4.04, 4.78	$\alpha = 0.220, \beta = 0.180$
65 - 70	4.48, 4.66, 4.22, 4.10, 4.88	$\alpha = 0.225, \beta = 0.180$
70 - 75	4.60, 4.56, 4.70, 4.28, 4.18	$\alpha = 0.225, \beta = 0.185$
75 - 80	4.66, 4.64, 4.22, 4.80, 4.62	$\alpha = 0.230, \beta = 0.185$
80 - 85	4.88, 4.80, 4.30, 4.88, 4.78	$\alpha = 0.230, \beta = 0.190$
85 - 90	4.90, 4.94, 4.40, 5.00, 4.90	$\alpha = 0.235, \beta = 0.190$
90 - 95	5.06, 5.16, 4.54, 5.12, 5.00	$\alpha = 0.235, \beta = 0.195$
95 - 100	5.14, 5.22, 4.60, 5.26, 5.08	$\alpha = 0.240, \beta = 0.195$

On each graph each heuristic's run occurs with a specific delay bound. Further, during each run the heuristic constructs multicast trees for 100 different multicast instances each of size 40 (40 node multicast destination sets). The average tree cost, averaged over the number of times (among the 100 multicast instances) in which a tree was successfully constructed, is computed and recorded. The delay bound values over the different runs are varied from 20 to 900 in increments of 20. Therefore average tree cost metric is measured and recorded 45 times on each graph and 4500 times over the 100 graphs. The total average tree cost of the heuristic is the average over the successful among the 4500 instances. Recall that all the heuristics have the same set of successes.

Tables 4.9, 4.10, and 4.11 compare the heuristics' performance with respect to DQDMR for the 100 graphs. The tables from the left to right (along the columns) denote the following:

1. *Heuristic(H):* The heuristic (H) that is being compared to DQDMR.

2. *Average C_H:* The average cost of multicast trees found by heuristic H ($\frac{Total\ C_H}{Times_H}$).

3. *Average C_{DQDMR}:* The average cost of multicast trees found by DQDMR ($\frac{Total\ C_{DQDMR}}{Times_{DQDMR}}$). As the average C_{DQDMR} value is the same along all the rows, this column is eliminated and the information is included in the table caption.

4. $\frac{C_H}{C_{DQDMR}}$*:* The ratio of the average cost of trees found by H over that found by DQDMR over all instances $\frac{\text{Average } C_H}{\text{Average } C_{DQDMR}}$.

The heuristics are arranged with the highest average tree cost heuristics listed at the top and the lowest average tree cost heuristics listed at the bottom.

4.3.1 EQUAL cost scenario

Table 4.9 shows the relative difference in the performance of the heuristics over 100 different Waxman graphs in the EQUAL cost scenario with respect to DQDMR. Listed below are the conclusions from the data shown in the table:

- DEPDT and DQDMR on an average construct multicast trees with the highest cost. Note that the ratio is $\frac{C_H}{C_{DQDMR}} = 1$ for DQDMR.

- DWH (at $k = 900$), ZWH, and the offline QDMR on an average construct multicast trees with lowest cost.

- The heuristics have the following performance order from worst to best in terms of tree cost over the 100 different graphs (DEPDT, DQDMR, DWH-020, UWH, DWH-100, TEPDT, TQDMR, DWH-160, DWH-300, DWH-900, ZWH, and QDMR)

- DWH (at $k \geq 160$) constructs on an average lower cost multicast trees than the best online variants of QDMR (TQDMR and TEPDT).

- The percentage difference in the performance of the best heuristic (offline QDMR) and the worst heuristic (DEPDT) is 36.66%.

- The percentage difference in the performance of DWH with respect to QDMR improves from 17.38% at $k = 100$ to 1.42% at $k = 900$. DWH with increasing k constructs multicast trees with lower cost.

4.3.2 REVERSE cost scenario

Table 4.10 shows the relative difference in the performance of the heuristics over 100 different Waxman graphs in the REVERSE cost scenario with respect to DQDMR. Listed below are the conclusions from the data shown in the table:

- DQDMR and DEPDT on an average construct multicast trees with the highest cost.

- DWH (at $k = 900$), and the offline QDMR on an average construct multicast trees with lowest cost.

- The heuristics have the following performance order from worst to best in terms of tree cost over the 100 different graphs DEPDT, DQDMR, UWH, DWH-020, DWH-100, DWH-160, TEPDT, DWH-300, TQDMR, ZWH, DWH-900, and QDMR

- DWH (at $k \geq 20$), UWH, and ZWH all on an average construct lower cost multicast trees than TQDMR and TEPDT (the best online variants of QDMR)

- The percentage difference in the performance of the best heuristic (offline QDMR) and the worst heuristic (DEPDT) is 28.84%.

- The percentage difference in the performance of DWH with respect to QDMR improves from 14.60% at $k = 100$ to 3.94% at $k = 900$. DWH with increasing k constructs multicast trees with lower cost.

4.3.3 RANDOM cost scenario

Table 4.11 shows the relative difference in the performance of the heuristics over 100 different Waxman graphs in the RANDOM cost scenario with respect to DQDMR. Listed below are the conclusions from the data shown in the table:

- DEPDT and DQDMR on an average construct multicast trees with the highest cost.

- ZWH, DWH (at $k = 900$, and $k = 300$), and the offline QDMR on an average construct multicast trees with lowest cost.

- The heuristics have the following performance order from worst to best in terms of tree cost over the 100 different graphs DEPDT, DQDMR, UWH, DWH-020, DWH-100, DWH-160, TEPDT, TQDMR, DWH-300, ZWH, DWH-900, and QDMR

- DWH (at $k \geq 20$), UWH, and ZWH all on an average construct lower cost multicast trees than TQDMR and TEPDT (the best online variants of QDMR)

Table 4.9: Heuristics' Cost Comparison wrt DQDMR over 100 graphs (EQUAL cost), Average C_{DQDMR}=1554.698

Heuristic(H)	Average C_H	$\frac{C_H}{C_{DQDMR}}$
DEPDT	1576.202	1.01383
DWH-020	1457.641	0.93757
UWH	1455.416	0.93614
DWH-100	1353.853	0.87081
TEPDT	1315.964	0.84644
TQDMR	1288.695	0.82890
DWH-160	1271.990	0.81816
DWH-300	1210.958	0.77890
DWH-900	1169.755	0.75240
ZWH	1160.927	0.74672
QDMR	1153.349	0.74185

- The percentage difference in the performance of the best heuristic (offline QDMR) and the worst heuristic (DEPDT) is 27.17%.

- The percentage difference in the performance of DWH with respect to QDMR improves from 13.21% at $k = 100$ to 1.55% at $k = 900$. DWH with increasing k constructs multicast trees with lower cost.

4.4 Comparison to the optimal solution:

In this section the heuristics' performance is compared to the optimal solution. The minimum cost delay constrained disjoint path multicast routing problem can be solved on small graphs through the use of exhaustive enumeration. A 10 node 16 edge graph is used in the comparison process. The exploration effort in exhaustively enumerating the 65536 edge subsets is reduced by using a filtration technique:

- Determine whether the edge subset forms a connected component.

Table 4.10: Heuristics' Cost Comparison wrt DQDMR over 100 graphs (REVERSE cost), Average C_{DQDMR}=1476.814

Heuristic(H)	Average C_H	$\frac{C_H}{C_{DQDMR}}$
DEPDT	1542.820	1.04469
UWH	1434.172	0.97113
DWH-020	1428.329	0.96717
DWH-100	1380.930	0.93507
DWH-160	1341.829	0.90860
TEPDT	1324.684	0.89699
DWH-300	1292.553	0.87523
TQDMR	1286.873	0.87138
ZWH	1265.853	0.85715
DWH-900	1252.517	0.84812
QDMR	1204.984	0.81593

Table 4.11: Heuristics' Cost Comparison wrt DQDMR over 100 graphs (RANDOM cost), Average C_{DQDMR}=1420.143

Heuristic(H)	Average C_H	$\frac{C_H}{C_{DQDMR}}$
DEPDT	1459.173	1.02748
UWH	1340.630	0.94401
DWH-020	1338.587	0.94257
DWH-100	1299.069	0.91474
DWH-160	1258.777	0.88637
TEPDT	1252.176	0.88173
TQDMR	1227.730	0.86451
DWH-300	1203.950	0.84777
DWH-900	1165.208	0.82049
ZWH	1164.339	0.81987
QDMR	1147.406	0.80795

- Determine whether the connected component has the source and the all the destination nodes.
- Determine whether the paths from the source to the destinations along the connected component meet the delay bound of the application.

The connected components which meet the above mentioned requirements are candidate multicast trees. From among these, select one which has the lowest tree cost. This multicast tree represents the lowest achievable cost (AC) for a delay-constrained minimum cost multicast tree.

Figure 4.19 shows the cumulative distribution of the *relative tree costs* obtained by the heuristics, compared to the optimal tree cost of the multicast tree constructed for 100 different multicast destination sets. Relative tree cost of a heuristic is the ratio of the cost of the tree constructed by the heuristic divided by AC. The cumulative distribution of the relative costs of the heuristics also shows a similar pattern as was observed in other evaluations. That is DWH (at high k values) and ZWH generally construct multicast trees with lower tree cost than the online heuristics. QDMR the offline multicast routing heuristic constructs multicast trees with a worst relative tree cost of only 120%. ZWH and DWH (at $k = 50$) have a worst relative tree cost of 1.4 which is better than the other online heuristics.

4.5 Summary

This chapter presents the evaluation of all the online, non-survivable, low-cost, delay-constrained multicast routing heuristics considered in this work. The evaluations are conducted in a structured manner and result in the following conclusions for all the three cost scenarios:

4.5.1 Evaluation of the heuristics on 4 Waxman graphs

- DQDMR and DEPDT construct higher cost multicast trees than all other heuristics.
- TQDMR and TEPDT construct lower cost multicast trees than DQDMR and DEPDT.
- DWH (at $k = 900$) constructs lower cost multicast trees than TQDMR.

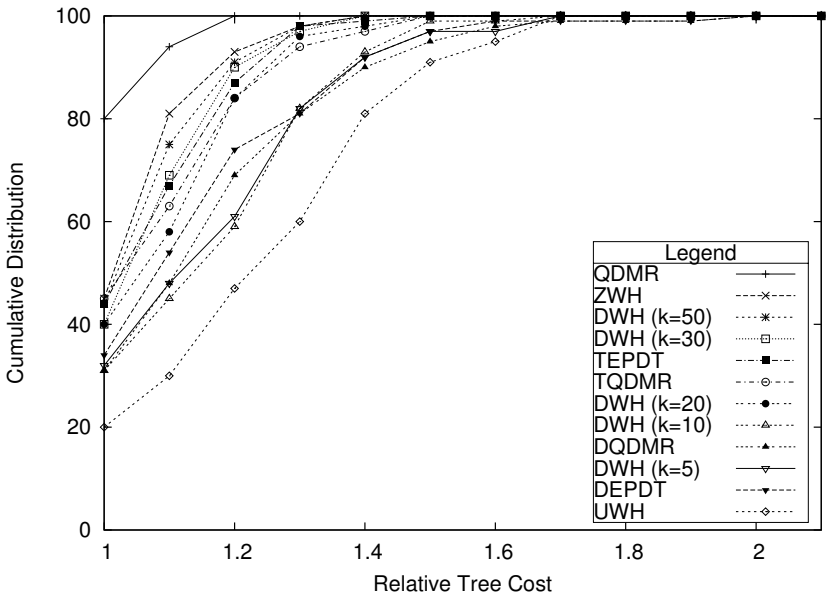

Figure 4.19: Cumulation distribution of the relative cost of the heuristics ($|N| = 10$, $|E| = 16$, $m = 6$, $\Delta = 50$)

- DWH (at $k = 900$), and ZWH construct lower cost multicast trees than all other online heuristics.

4.5.2 Evaluation on disparate graphs

- TQDMR and TEPDT construct lower cost multicast trees than DQDMR and DEPDT.
- DWH with increasing k constructs lower cost multicast trees.

4.5.3 Evaluations on 100 different Waxman graphs

- DWH (at $k = 900$) and ZWH on an average construct lower cost multicast trees than all other online heuristics.
- DWH (at $k = 900$) constructs multicast trees with 10%, 3%, and 5% lower cost than the best node-priority based heuristic TQDMR for EQUAL, REVERSE, and RANDOM cost scenarios respectively.

- TQDMR constructs multicast trees with 21%, 15%, and 16% lower cost than DQDMR for EQUAL, REVERSE, and RANDOM cost scenarios respectively.

4.5.4 Comparison to the optimal solution

- ZWH and DWH (at $k = 50$) construct multicast trees with a worst relative tree cost of 1.4 (compared to the optimal solution) which is better than the other online heuristics.

CHAPTER FIVE

EVALUATION OF SURVIVABLE LOW-COST DELAY-CONSTRAINED MULTICAST ROUTING HEURISTICS

This chapter presents the evaluation of the survivable multicast routing heuristics (DQDMR-DPP, DEPDT-DPP, TQDMR-DPP, TEPDT-DPP, ZW-DPP, UW-DPP, and DW-DPP) which construct node disjoint path pairs (DPP) to each destination node with low DPP graph cost while meeting the delay requirements on paths to destinations. Hence the relevant evaluation metric average DPP graph cost is selected to compare the performance of the heuristics at various delay bound values.

The heuristics as done in Chapter 4 are evaluated extensively and in a structured manner.

- *Evaluation of the heuristics on 4 Waxman graphs:* In Section 5.1 the heuristics are all evaluated on 4 different Waxman, [14], graphs with average node degrees (5.86, 6.86, 7.88, and 8.86) covering a range of graphs of the same type but with varying connectivity.

- *Evaluation on disparate graphs:* In Section 5.2 the heuristics are compared on different graphs. The graphs used in this section include 10 different Waxman graphs, a Pure Random graph, a Locality graph, and a Transit-Stub graph. This section will help answer the question of whether the type of the graph used in the evaluation affects the heuristics performance.

- *Evaluations on 100 different Waxman Graphs:* In Section 5.3 the heuristics are compared on 100 different 100 node Waxman graphs. The average node degree of the graphs used in this section varies from a low of 5.16 to a high of 8.86. Note that these graphs have on an average higher node degree than the ones used in Chapter 4, that is because node disjoint paths in a graph are only possible when the nodes in the graph are doubly-connected. The average node degrees of all the 100 graphs used in the evaluation are listed in the Table 5.9.

- *Comparison to the Optimal Solution:* In Section 5.4 the heuristics' performance is compared to the optimal solution on a 10 node 16 edge graph.

Table 5.1: Heuristic running times (in seconds) on a Waxman graph:100 Nodes, Degree=5.86, $\alpha = 0.245, \beta = 0.210$

Heuristic	Running Time
DW-DPP ($k = 20$)	0.0169
DW-DPP ($k = 100$)	0.0171
DW-DPP ($k = 160$)	0.0172
DW-DPP ($k = 300$)	0.0174
DW-DPP ($k = 900$)	0.0178
UW	0.0168
ZW	0.0181
DQDMR-DPP	0.0185
DEPDT-DPP	0.0185
TQDMR-DPP	0.0204
TEPDT-DPP	0.0205

Other evaluation criterion which are of interest in the evaluation of multicast routing heuristics are the evaluation of the heuristics based on the computational complexity and the running time of the heuristics. Computational complexity has not been considered as an evaluation metric in this research work because of the fact that all the heuristics have the same computational complexity as stated in Chapter 3.

The running times of all the heuristics are also similar as illustrated in Table 5.1. The heuristics are all run on an Intel Pentium 4 machine with a clock speed of 2.4 GHz and with 512 MB of RAM. The running times listed in the table are the averages over the successful among the 100 multicast graph construction instances when the heuristics are executed on a 100 node Waxman graph with average node degree 5.76 (graph construction parameters, $\alpha = 0.245$ and $\beta = 0.21$). As shown in the table the running times of the heuristics are similar and are in the range [0.0168, 0.0205] seconds.

5.1 Evaluation of the heuristics on 4 Waxman graphs

The heuristics all construct low-cost disjoint path pairs to each destination, therefore the average cost of DPP graph computed over the 100 multicast sessions is used as the performance metric and as a data point in the graphs. The simulation set-up is similar to that used for non-survivable heuristics in Chapter 4.

Figure 5.1 compares the heuristics' DPP graph cost performance for varying delay bound values when run on a 100 Node Waxman graph (Degree=5.86, $\alpha = 0.245, \beta = 0.210$) for the EQUAL cost scenario. From the figure we observe:

- DW-DPP (at $k = 900$) and ZW-DPP construct DPP graphs with lower cost than all other heuristics. While DEPDT-DPP and DQDMR-DPP construct DPP graphs with higher cost than all other heuristics.

- The cost of the DPP graphs constructed by DW-DPP decreases with increasing k. DW-DPP at low k (< 100) values constructs DPP graphs with similar cost to its least-cost-path DPP variant UW-DPP and at high k (≥ 300) constructs DPP graphs with costs similar to its least graph cost DPP variant ZW-DPP. Also DW-DPP at $k \geq 160$ constructs DPP graphs with lower cost than the tree node priority influenced DPP variants of QDMR, TQDMR-DPP and TEPDT-DPP.

- However, TQDMR-DPP and TEPDT-DPP construct lower cost DPP graphs than the destination node priority influenced DPP variants of QDMR, DQDMR-DPP and DEPDT-DPP. Considering the tree nodes which are more in number than the destination nodes helps in producing lower cost DPP graphs.

Figure 5.2 compares the heuristics on the same graph but for REVERSE cost scenario. In this figure the node priority influenced heuristics (DQDMR-DPP variants) construct lower cost DPP graphs than the edge priority influenced heuristics (DW-DPP variants) at lower delay bound

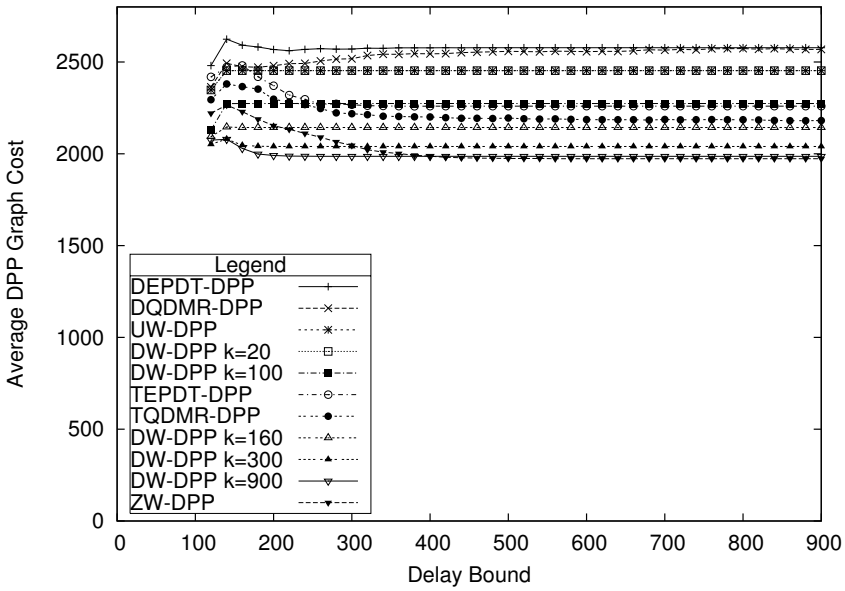

Figure 5.1: DPP graph cost versus Δ at $m = 40$ (EQUAL cost) Waxman graph: 100 Nodes, Degree=5.86, $\alpha = 0.245$, $\beta = 0.210$

values (< 400) but higher cost DPP graphs at higher delay bound values. However with increasing delay bound values the heuristics' performance is comparable to that observed in the EQUAL cost scenario.

Figure 5.3 compares the heuristics for the RANDOM cost scenario. We observe that the heuristics DPP graph cost performance is similar to that observed in EQUAL cost case with minor modifications.

It is also interesting to note that in all the three cost scenarios, ZW-DPP constructs higher cost DPP graphs at lower delay bound values and lower cost DPP graphs (close to that constructed by DW-DPP at $k = 900$) at higher delay bound values. A reason for this can be the delay-insensitivity inherent in ZW-DPP. At lower delay bound values the lower total cost focussed ZW-DPP fails initially to meet the delay bound in the first pass of the heuristic and is forced to add the least delay path pairs to that destination to the DPP graph adding to the cost of the final DPP graph.

Figures 5.4 - 5.12 illustrate similar differences for three other Waxman graphs with degrees (6.86, 7.88, and 8.86).

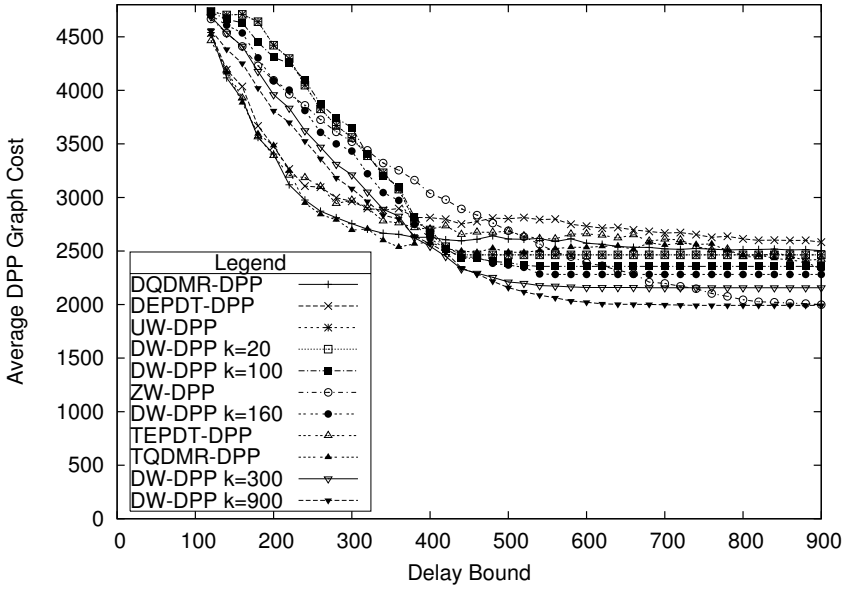

Figure 5.2: DPP graph cost versus Δ at $m = 40$ (REVERSE cost)
Waxman graph: 100 Nodes, Degree=5.86, $\alpha = 0.245$, $\beta = 0.210$

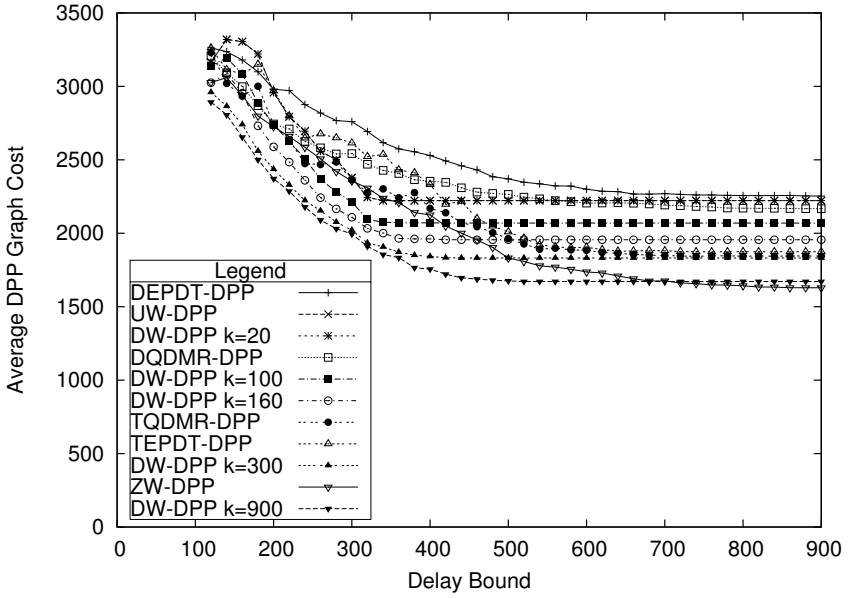

Figure 5.3: DPP graph cost versus Δ at $m = 40$ (RANDOM cost)
Waxman graph: 100 Nodes, Degree=5.86, $\alpha = 0.245$, $\beta = 0.210$

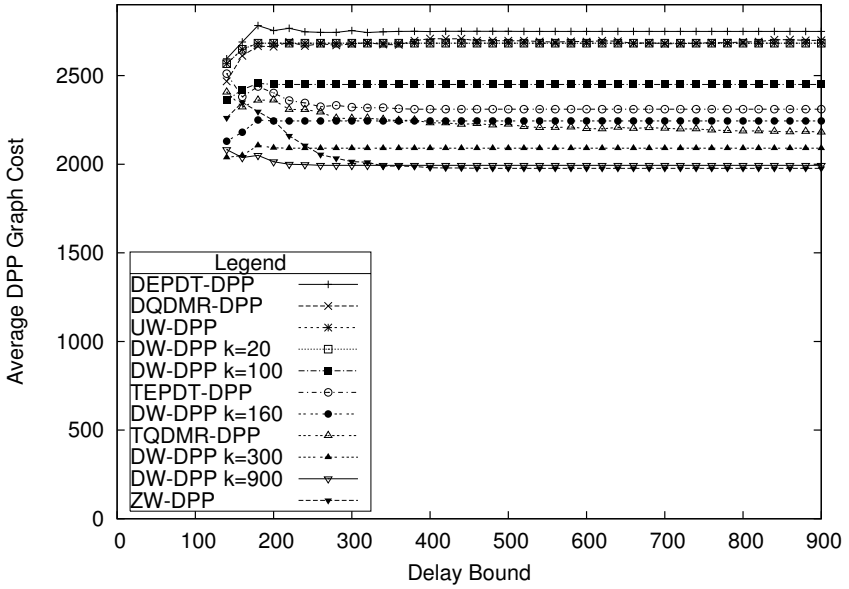

Figure 5.4: DPP graph cost versus Δ at $m = 40$ (EQUAL cost) Waxman graph: 100 Nodes, Degree=6.86, $\alpha = 0.265$, $\beta = 0.225$

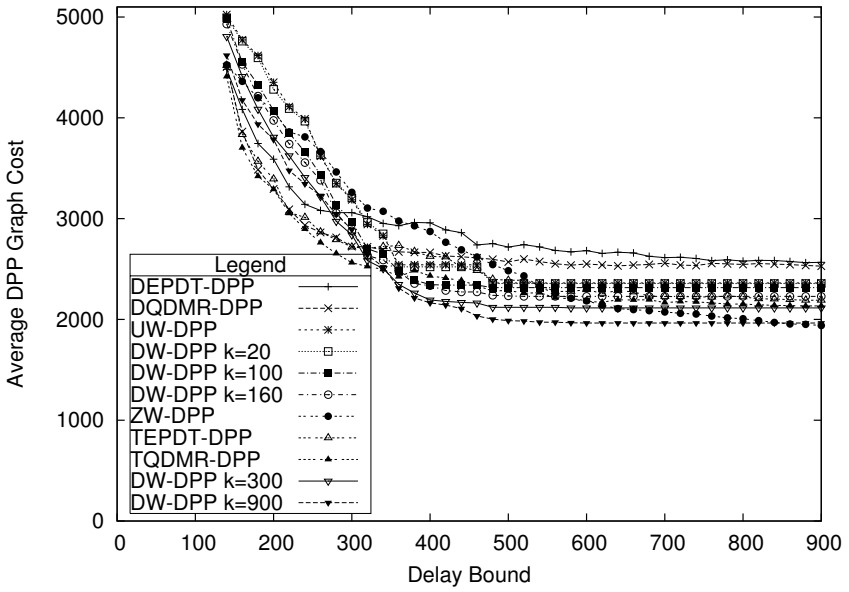

Figure 5.5: DPP graph cost versus Δ at $m = 40$ (REVERSE cost) Waxman graph: 100 Nodes, Degree=6.86, $\alpha = 0.265$, $\beta = 0.225$

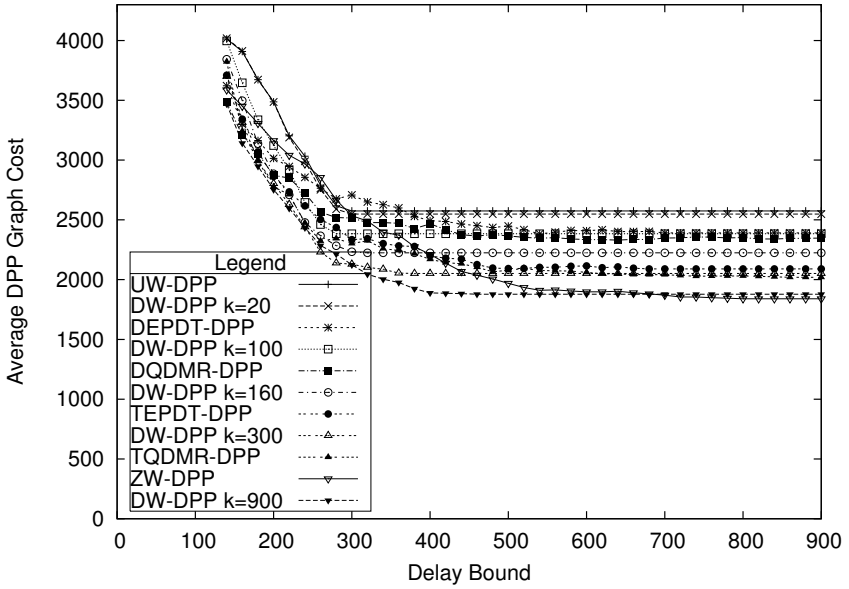

Figure 5.6: DPP graph cost versus Δ at $m = 40$ (RANDOM cost) Waxman graph: 100 Nodes, Degree=6.86, $\alpha = 0.265$, $\beta = 0.225$

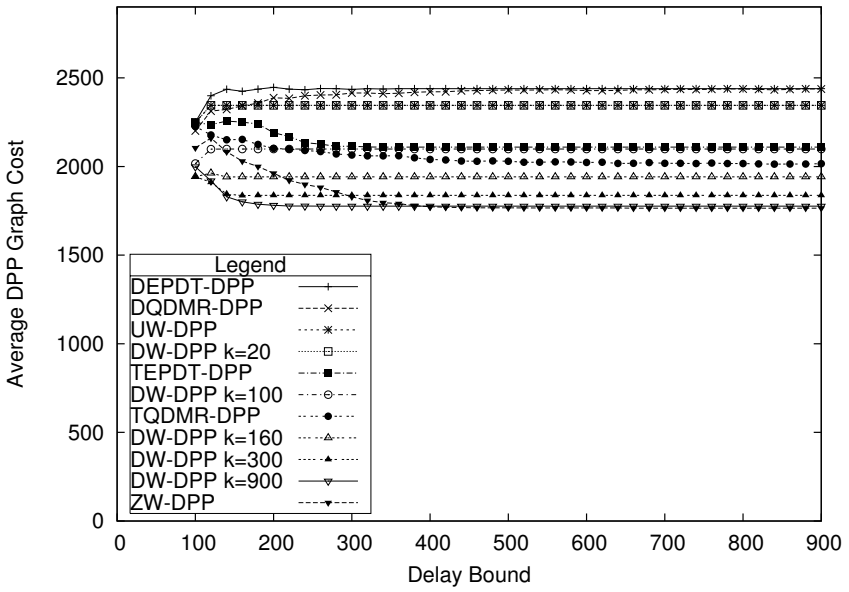

Figure 5.7: DPP graph cost versus Δ at $m = 40$ (EQUAL cost) Waxman graph: 100 Nodes, Degree=7.88, $\alpha = 0.280$, $\beta = 0.245$

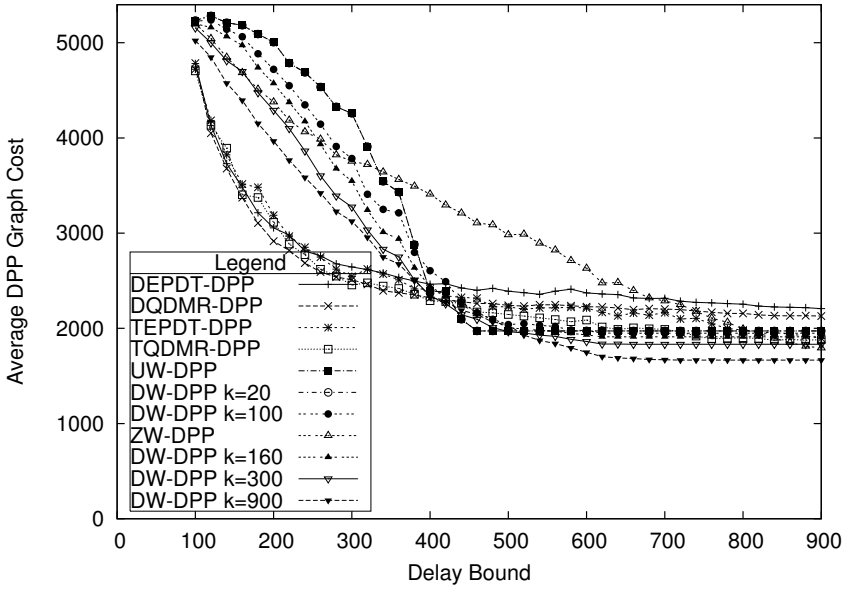

Figure 5.8: DPP graph cost versus Δ at $m = 40$ (REVERSE cost)
Waxman graph: 100 Nodes, Degree=7.88, $\alpha = 0.280$, $\beta = 0.245$

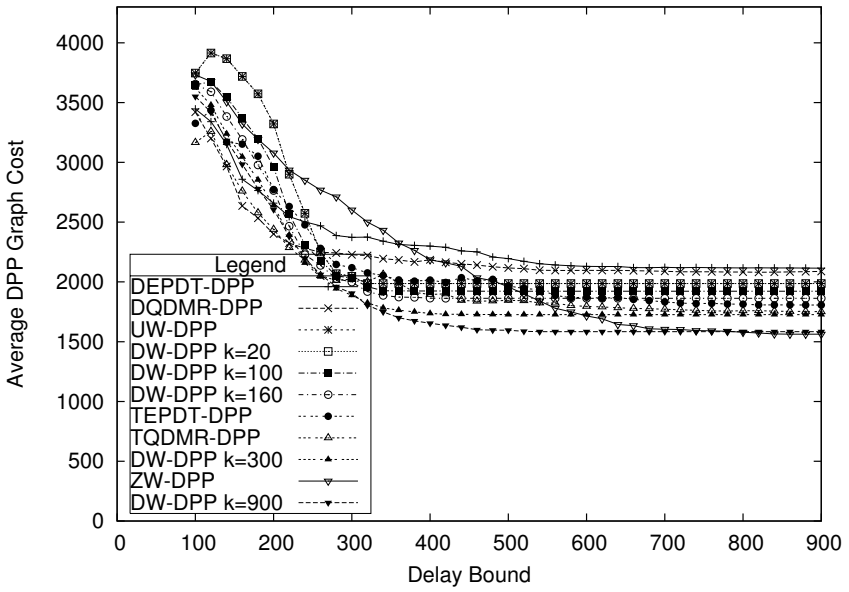

Figure 5.9: DPP graph cost versus Δ at $m = 40$ (RANDOM cost)
Waxman graph: 100 Nodes, Degree=7.88, $\alpha = 0.280$, $\beta = 0.245$

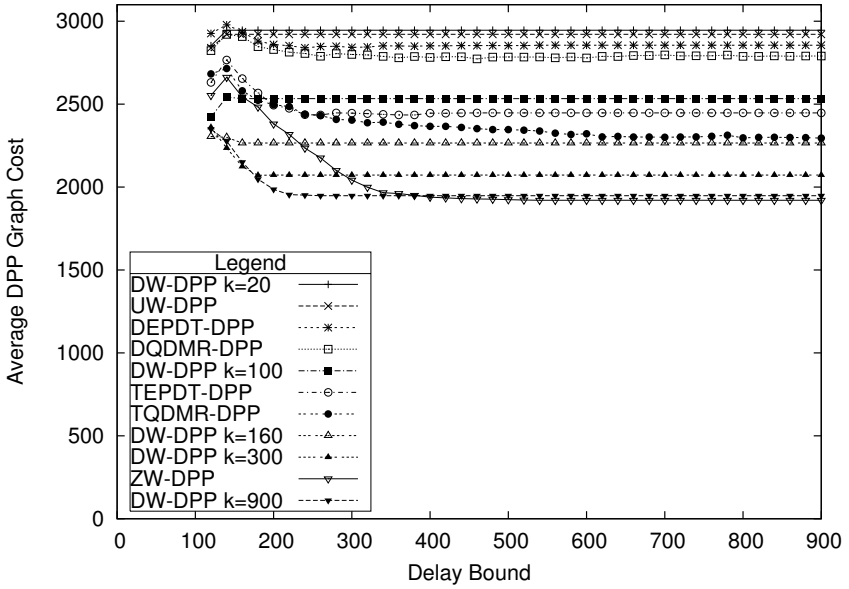

Figure 5.10: DPP graph cost versus Δ at $m = 40$ (EQUAL cost)
Waxman graph: 100 Nodes, Degree=8.86, $\alpha = 0.295$, $\beta = 0.255$

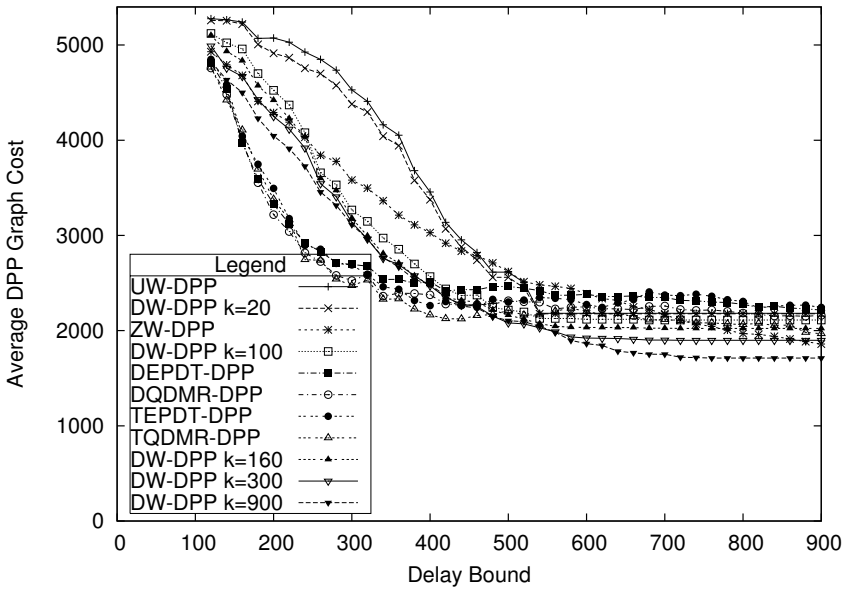

Figure 5.11: DPP graph cost versus Δ at $m = 40$ (REVERSE cost)
Waxman graph: 100 Nodes, Degree=8.86, $\alpha = 0.295$, $\beta = 0.255$

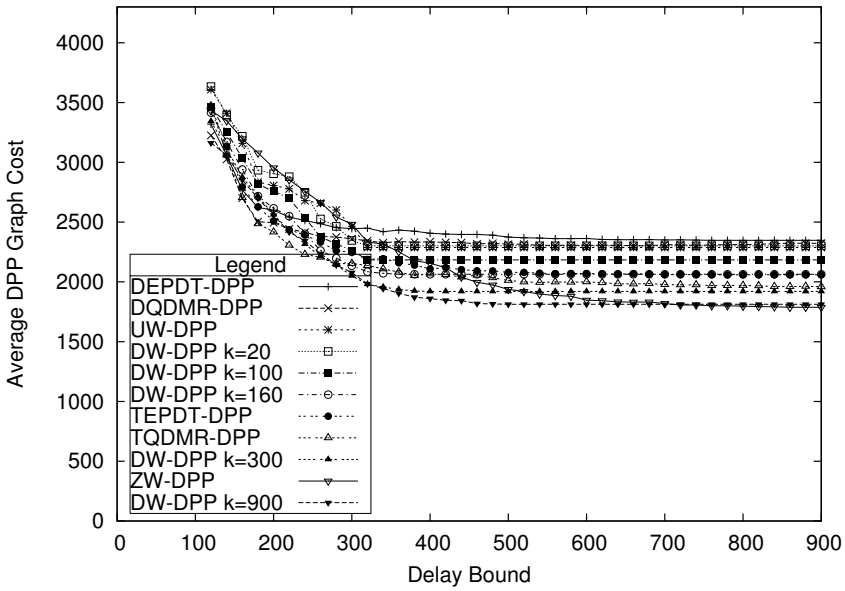

Figure 5.12: DPP graph cost versus Δ at $m = 40$ (RANDOM cost)
Waxman graph: 100 Nodes, Degree=8.86, $\alpha = 0.295$, $\beta = 0.255$

5.2 Evaluation on disparate graphs

As the performance of a multicast routing heuristic is dependant on the graph used in the evaluation process, to get a better understanding of the heuristics' performance on different graphs the evaluations in this section are performed on ten 100 node Waxman graphs. The properties of the Waxman graphs used in this section are listed in Table 5.2. Apart from these Waxman graphs the evaluations were also conducted on:

- *Pure Random* A 100 node Pure Random graph with average node degree 5.98. This random graph was generated with the probability ($p = 0.063$)

- *Locality* A 100 node Locality graph (average node degree = 5.96), generated with the following parameters ($\alpha = 0.062$, $\beta = .005$, and $C = 35.00$).

- *Transit-Stub* A 100 node Transit-Stub graph with average node degree 7.94. The Transit-Stub graph has 1 transit domain of 4 nodes, 3 stub domains for each transit node and an average of 8 nodes per stub domain. The higher average node degree in the Transit-Stub

graph is to account for the poor connectivity between the different domains. Even at this node-degree some of the nodes in the graph are singly connected.

The performance of the heuristics over the different graphs listed above and for the EQUAL cost scenario is shown in Table 5.3. The evaluation results for the other two cost scenarios are shown in Table 5.5 (REVERSE cost) and Table 5.7 (RANDOM cost). In these tables the heuristics' performance order for a particular graph is listed in descending order along a row as done in Chapter 4. Note that in the tables the heuristics' names are abbreviated to account for the limited space in the table. The heuristics used in the evaluation are the DPP variants of the heuristics listed.

Average DPP graph cost is the average over all the DPPs constructed for each graph used in the evaluation. For each heuristic at each delay bound value the average DPP graph cost is the average cost of the DPP graphs constructed for the 100 multicast destination sets that are fed at that delay bound value. The delay bound is varied from 20 to 900 in increments of 20 to account for 4500 DPP graph construction instances.

To help in the relative performance assessment, the heuristics are divided into two groups:

1. *DQDMR-DPP Group* These heuristics are the online DPP variants of QDMR. These are the destination node priority influenced DPP variants, DQDMR-DPP and DEPDT-DPP, and the tree node priority influenced DPP variants, TQDMR-DPP and TEPDT-DPP.

2. *DW-DPP Group* These heuristics are DPP variants of DWH and its related heuristics. The heuristics that are part of this group are DW-DPP, UW-DPP and ZW-DPP.

Tables 5.4, 5.6, and 5.8 present an alternative illustration of the average cost of DPP graphs constructed by the heuristics. The heuristics are arranged in fixed columns for a better illustration of the performance difference between the two groups. The DQDMR-DPP group heuristics are listed on the left side of the table and the DW-DPP based heuristics are listed on the right side of the table.

Figures. 5.13 - 5.18 illustrate the tree cost performance of the two different heuristics groups for EQUAL, REVERSE, and RANDOM cost scenarios. The figures plot the low and high values of the average DPP graph cost observed in each heuristic group for each kind of graph.

In order to improve the presentation clarity the performance of the 13 different graphs for each cost scenario is presented in two graphs. The first graph in each case presents the performance of the heuristics for graphs W1 (short for Waxman1), W2, W3, W4, W5, and W6 (Disparate Graphs-I). The second graph in each case presents the performance of the heuristics for graphs W7, W8, W9, W10, Random (Pure Random), Locality, and TS (Transit-Stub) (Disparate Graphs-II).

5.2.1 EQUAL cost scenario

From Table 5.3 we see that the graph used in the evaluation process affects the performance of the heuristic. These general conslusions can be drawn from the data in the table:

- DW-DPP (at $k = 900$) consistently constructs the lowest cost DPP graphs. DW-DPP (at k=300 and at $k = 900$) together with ZW-DPP construct lower cost DPP graphs than all other heuristics.
- The DEPDT heuristic usually constructs higher cost DPP graphs than all other heuristics.
- The tree node priority influenced DPP variants of QDMR, (TQDMR-DPP and TEPDT-DPP) generally construct lower cost DPP graphs than the more restrictive destination node priority influenced DPP variants of QDMR (DQDMR-DPP and DEPDT-DPP).

For the six Waxman graphs (W1 to W6) listed in Figure 5.13 (Disparate Graphs-I) it is interesting to note that the low value of the DPP graph cost for DW-DPP group heuristics is lower than that for DQDMR-DPP group heuristics. Also the high value of the DPP graph cost for DW-DPP group heuristics is usually lower than the high value for the DQDMR-DPP group heuristics.

Similarly for the rest of the graphs (Disparate Graphs-II) shown in Figure 5.14, the low-high bar columns show that the low value of the DPP graph cost for DW-DPP group heuristics is always

Table 5.2: Ten Waxman Graphs

Name	Average Node Degree	Graph Parameters
Waxman1	6.14	$\alpha = 0.250, \beta = 0.215$
Waxman2	5.94	$\alpha = 0.250, \beta = 0.215$
Waxman3	5.50	$\alpha = 0.250, \beta = 0.215$
Waxman4	6.12	$\alpha = 0.250, \beta = 0.215$
Waxman5	6.10	$\alpha = 0.250, \beta = 0.215$
Waxman6	7.36	$\alpha = 0.270, \beta = 0.235$
Waxman7	6.82	$\alpha = 0.270, \beta = 0.235$
Waxman8	6.66	$\alpha = 0.270, \beta = 0.235$
Waxman9	7.50	$\alpha = 0.270, \beta = 0.235$
Waxman10	7.22	$\alpha = 0.270, \beta = 0.235$

lower than the low value of the DPP graph cost for the DQDMR-DPP group heuristics. However the high value of the DPP graph cost for the DW-DPP group heuristics is sometimes higher than the high value of DPP graphs constructed by the DQDMR-DPP group heuristics. UW-DPP and DW-DPP (at low k) construct higher cost DPP graphs as they are more inclined towards improving their delay performance than their DPP graph cost performance.

5.2.2 REVERSE cost scenario

From the Table 5.5 (REVERSE cost scenario) we again observe that the graph used in the evaluation process influences the relative performance of the heuristics. However, the general conclusions that can be drawn from the data shown in the table are the following :

- DW-DPP ($k = 900$) consistently constructs lower cost DPP graphs than all other heuristics.
- The tree node priority influenced DPP variants of QDMR, (TQDMR-DPP and TEPDT-DPP) generally construct DPP graphs with lower cost than DQDMR-DPP and DEPDT-DPP.
- UW-DPP along with DEPDT-DPP constructs the highest cost DPP graphs.

Table 5.3: Heuristics' Average DPP Graph Cost Performance Order Over Different Graphs (EQUAL cost)

	H	DEPDT	DQDMR	UW	DW-020	TEPDT	DW-100	TQDMR	DW-160	DW-300	ZW	DW-900
Waxman1	Cost	2478.46	2478.38	2411.16	2411.16	2239.53	2217.51	2191.63	2098.99	2006.70	1998.42	1957.61
	H	DW-020	UW	DEPDT	DQDMR	DW-100	TEPDT	TQDMR	DW-160	DW-300	ZW	DW-900
Waxman2	Cost	2898.06	2851.38	2753.78	2700.21	2482.18	2436.67	2332.44	2276.51	2131.45	2077.97	2049.55
	H	DEPDT	DQDMR	DW-020	UW	DW-100	TEPDT	DW-160	TQDMR	DW-300	ZW	DW-900
Waxman3	Cost	2928.17	2836.00	2822.11	2790.24	2615.87	2460.42	2411.84	2405.85	2199.49	2147.93	2089.52
	H	DEPDT	DQDMR	UW	DW-020	DW-100	TEPDT	DW-160	TQDMR	DW-300	ZW	DW-900
Waxman4	Cost	2778.02	2706.86	2648.63	2648.63	2537.20	2384.50	2366.78	2311.74	2177.95	2094.70	2087.45
	H	UW	DW-020	DEPDT	DQDMR	DW-100	TEPDT	TQDMR	DW-160	DW-300	ZW	DW-900
Waxman5	Cost	2710.43	2701.71	2626.41	2616.99	2428.59	2423.44	2328.74	2273.28	2145.71	2085.23	2065.60
	H	DEPDT	DQDMR	UW	DW-020	TEPDT	DW-100	TQDMR	DW-160	DW-300	ZW	DW-900
Waxman6	Cost	2445.98	2421.00	2346.93	2346.93	2140.28	2102.98	2065.55	1965.72	1862.06	1838.59	1799.96
	H	DW-020	UW	DEPDT	DQDMR	TEPDT	DW-100	TQDMR	DW-160	DW-300	ZW	DW-900
Waxman7	Cost	2866.60	2818.45	2719.38	2668.51	2397.43	2396.57	2281.26	2203.55	2054.09	2015.50	1985.66
	H	DEPDT	DW-020	UW	DQDMR	DW-100	TEPDT	DW-160	TQDMR	DW-300	ZW	DW-900
Waxman8	Cost	2878.50	2816.82	2803.70	2798.49	2559.02	2420.71	2334.43	2330.96	2118.76	2055.77	1998.10
	H	DW-020	UW	DEPDT	DQDMR	DW-100	TEPDT	TQDMR	DW-160	DW-300	ZW	DW-900
Waxman9	Cost	2976.59	2975.75	2854.23	2818.35	2581.89	2450.44	2364.73	2294.79	2085.06	2029.29	1983.62
	H	DEPDT	UW	DW-020	DQDMR	DW-100	TEPDT	DW-160	TQDMR	DW-300	ZW	DW-900
Waxman10	Cost	2688.87	2685.70	2685.70	2635.31	2415.07	2272.13	2194.79	2183.52	2033.57	1971.70	1940.53
	H	DEPDT	DQDMR	UW	DW-020	DW-100	TEPDT	DW-160	TQDMR	DW-300	ZW	DW-900
Random	Cost	4107.48	4014.82	3883.40	3883.40	3787.53	3637.52	3596.59	3524.93	3334.92	3292.52	3196.92
	H	DEPDT	DQDMR	UW	DW-020	DW-100	TEPDT	DW-160	TQDMR	DW-300	ZW	DW-900
Locality	Cost	4106.30	4014.18	3883.40	3883.40	3787.53	3638.33	3596.19	3524.44	3334.66	3292.53	3197.01
	H	DEPDT	DW-020	UW	DQDMR	TEPDT	TQDMR	DW-100	DW-160	ZW	DW-300	DW-900
TS	Cost	683.63	682.92	682.70	680.12	646.73	646.70	611.67	607.49	599.26	599.22	596.30

Table 5.4: Heuristics' Average DPP Graph Cost Performance Over Different Graphs (EQUAL cost)

	DEPDT	DQDMR	TEPDT	TQDMR	UW	DW-020	DW-100	DW-160	DW-300	ZW	DW-900
Waxman1	2478.46	2478.38	2239.53	2191.63	2411.16	2411.16	2217.51	2098.99	2006.70	1998.42	1957.61
Waxman2	2753.78	2700.21	2436.67	2332.44	2851.38	2898.06	2482.18	2276.51	2131.45	2077.97	2049.55
Waxman3	2928.17	2836.00	2460.42	2405.85	2790.24	2822.11	2615.87	2411.84	2199.49	2147.93	2089.52
Waxman4	2778.02	2706.86	2384.50	2311.74	2648.63	2648.63	2537.20	2366.78	2177.95	2094.70	2087.45
Waxman5	2626.41	2616.99	2423.44	2328.74	2710.43	2701.71	2428.59	2273.28	2145.71	2085.23	2065.60
Waxman6	2445.98	2421.00	2140.28	2065.55	2346.93	2346.93	2102.98	1965.72	1862.06	1838.59	1799.96
Waxman7	2719.38	2668.51	2397.43	2281.26	2818.45	2866.60	2396.57	2203.55	2054.09	2015.50	1985.66
Waxman8	2878.50	2798.49	2420.71	2330.96	2803.70	2816.82	2559.02	2334.43	2118.76	2055.77	1998.10
Waxman9	2854.23	2818.35	2450.44	2364.73	2975.75	2976.59	2581.89	2294.79	2085.06	2029.29	1983.62
Waxman10	2688.87	2635.31	2272.13	2183.52	2685.70	2685.70	2415.07	2194.79	2033.57	1971.70	1940.53
Random	4107.48	4014.82	3637.52	3524.93	3883.40	3883.40	3787.53	3596.59	3334.92	3292.52	3196.92
Locality	4106.30	4014.18	3638.33	3524.44	3883.40	3883.40	3787.53	3596.19	3334.66	3292.53	3197.01
TS	683.63	680.12	646.73	646.70	682.70	682.92	611.67	607.49	599.22	599.26	596.30

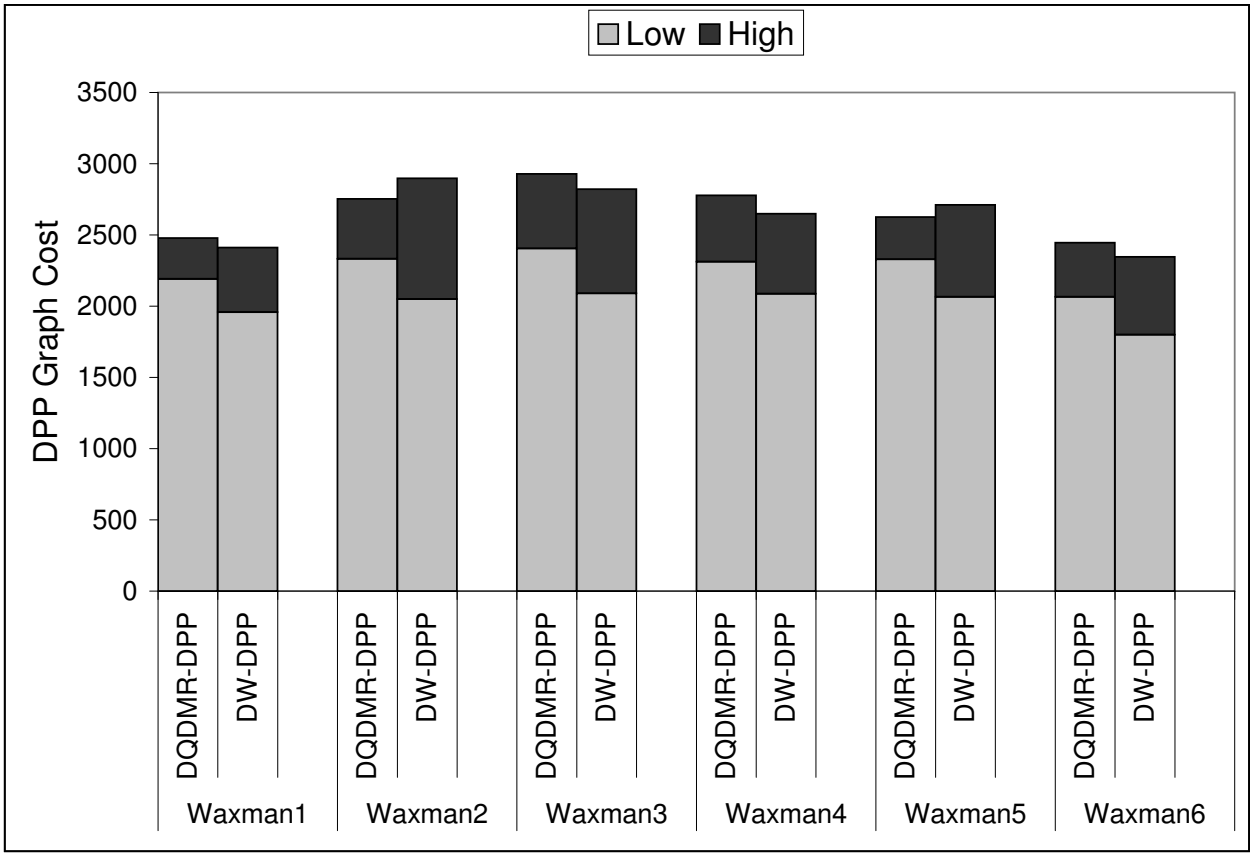

Figure 5.13: Heuristics Performance on Disparate Graphs-I (EQUAL cost)

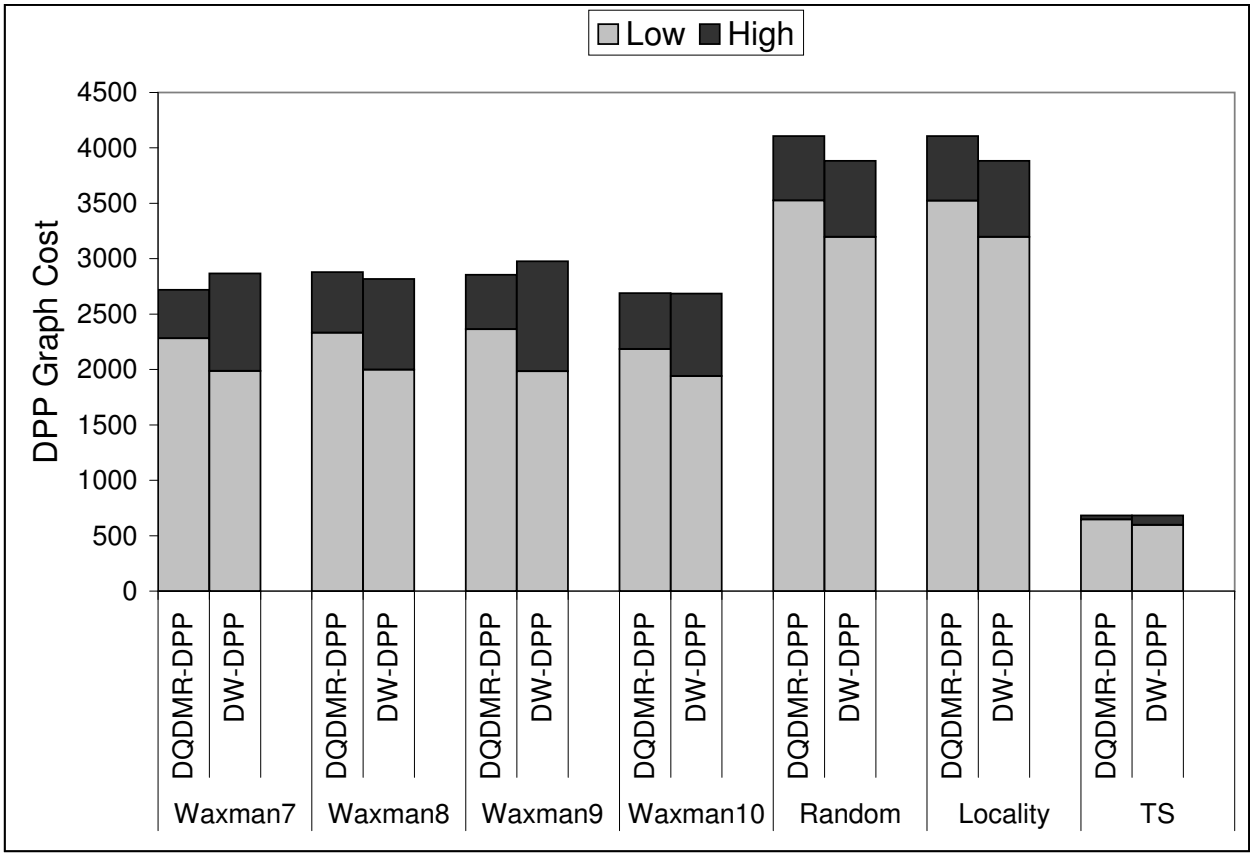

Figure 5.14: Heuristics Performance on Disparate Graphs-II (EQUAL cost)

From the disparate graphs listed in Figure 5.15 and Figure 5.16 it is evident that the low value of the DPP graph cost for DW-DPP and related heuristics is lower than or close to the low value of the DPP graph cost for DQDMR-DPP and related heuristics. Conversely the high value of the DPP graph cost is higher for DW-DPP group than for DQDMR-DPP group. This is because UW-DPP and DW-DPP (at $k < 100$) on some occasions construct higher cost DPP graphs.

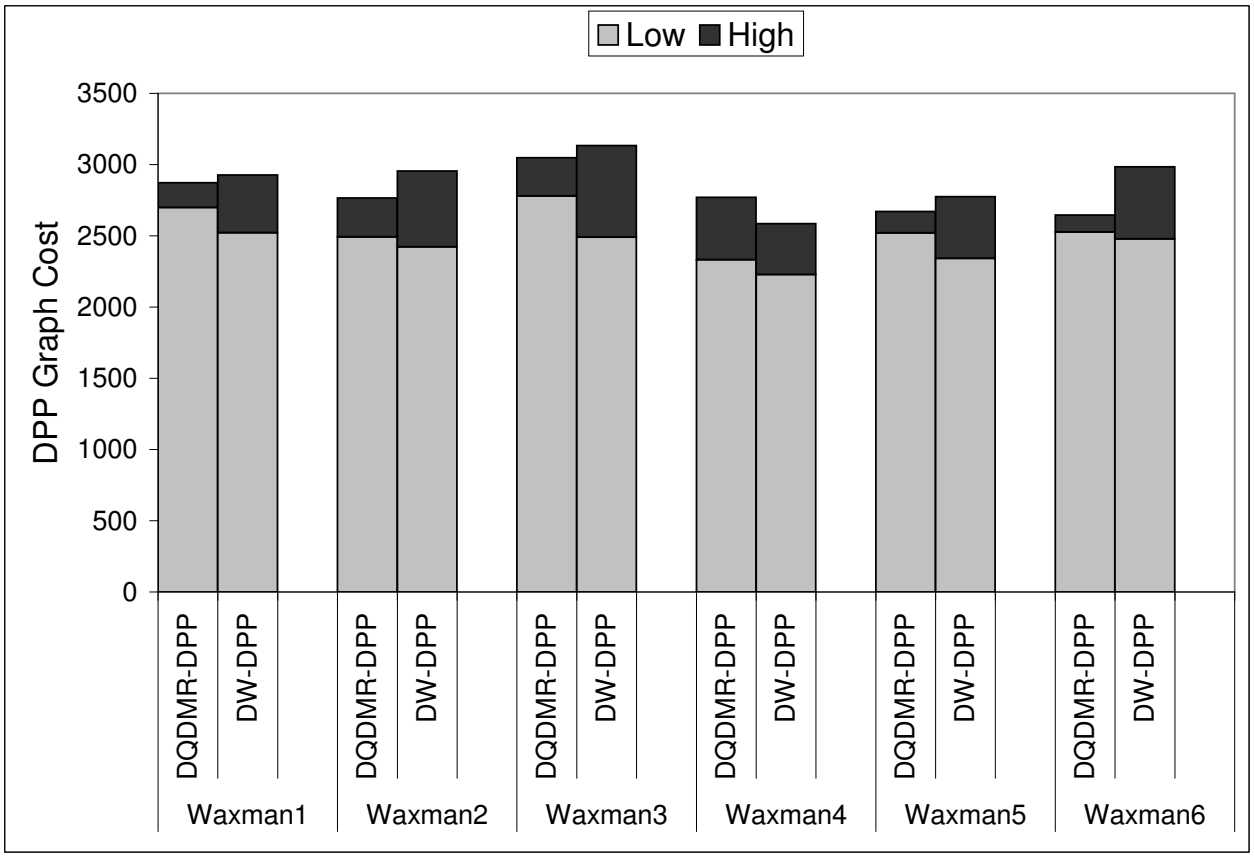

Figure 5.15: Heuristics Performance on Disparate Graphs-I (REVERSE cost)

5.2.3 RANDOM cost scenario

As noted earlier the relative performance of the heuristics is influenced by the graph used in the evaluations as is evidenced in Table 5.7 (RANDOM cost scenario). However, the following general conclusions are drawn from the data shown in the table:

Table 5.5: Heuristics' Average DPP Graph Cost Performance Order Over Different Graphs (REVERSE cost)

	H	ZW	DEPDT	DW-020	UW	DW-100	TEPDT	DW-160	DQDMR	TQDMR	DW-300	DW-900
Waxman1	Cost	2927.60	2873.07	2865.36	2865.36	2860.39	2810.03	2782.18	2714.29	2698.84	2625.72	2522.44
	H	UW	DW-020	DW-100	DEPDT	TEPDT	DW-160	DQDMR	ZW	DW-300	TQDMR	DW-900
Waxman2	Cost	2955.12	2870.65	2801.29	2765.09	2727.16	2677.73	2670.43	2625.60	2536.54	2492.74	2422.16
	H	UW	DEPDT	DW-020	TEPDT	DW-100	ZW	DQDMR	DW-160	TQDMR	DW-300	DW-900
Waxman3	Cost	3133.30	3047.40	3046.04	2957.87	2915.06	2914.56	2859.94	2811.33	2779.46	2626.31	2491.22
	H	DEPDT	DQDMR	UW	DW-020	ZW	DW-100	DW-160	TEPDT	TQDMR	DW-300	DW-900
Waxman4	Cost	2770.25	2644.18	2585.10	2583.52	2490.75	2488.20	2436.79	2409.31	2332.58	2299.71	2228.77
	H	UW	DW-020	ZW	TEPDT	DEPDT	DW-100	DQDMR	DW-160	TQDMR	DW-300	DW-900
Waxman5	Cost	2774.67	2727.52	2713.02	2670.46	2637.77	2581.18	2533.50	2524.40	2519.99	2440.66	2341.41
	H	ZW	DW-020	UW	DW-100	DW-160	DEPDT	TEPDT	DW-300	DQDMR	TQDMR	DW-900
Waxman6	Cost	2984.69	2903.71	2903.71	2847.02	2741.47	2645.78	2615.62	2587.52	2528.21	2526.66	2477.91
	H	UW	DW-020	DW-100	DEPDT	ZW	DW-160	DQDMR	TEPDT	DW-300	TQDMR	DW-900
Waxman7	Cost	3006.50	2962.37	2900.89	2819.81	2790.33	2786.43	2699.72	2685.14	2645.86	2575.11	2524.41
	H	UW	ZW	DW-020	DW-100	DEPDT	DW-160	DQDMR	TEPDT	DW-300	TQDMR	DW-900
Waxman8	Cost	3290.05	3255.82	3218.48	3073.49	3007.42	2936.38	2867.03	2840.93	2741.94	2709.23	2617.03
	H	UW	DW-020	DW-100	ZW	TEPDT	DW-160	DEPDT	DQDMR	DW-300	TQDMR	DW-900
Waxman9	Cost	3083.28	3057.27	2707.57	2693.68	2592.29	2584.10	2580.90	2472.74	2469.74	2398.00	2393.84
	H	DEPDT	DW-020	UW	ZW	DQDMR	DW-100	DW-160	TEPDT	DW-300	TQDMR	DW-900
Waxman10	Cost	2774.31	2671.73	2667.46	2640.44	2605.03	2564.44	2487.12	2446.98	2367.39	2365.98	2284.65
	H	DEPDT	ZW	DW-020	UW	DQDMR	DW-100	TEPDT	DW-160	DW-300	TQDMR	DW-900
Random	Cost	4811.86	4649.50	4635.88	4635.88	4568.55	4497.52	4461.51	4448.40	4291.57	4276.13	4060.45
	H	DEPDT	ZW	DW-020	UW	DQDMR	DW-100	DW-160	TEPDT	DW-300	TQDMR	DW-900
Locality	Cost	4788.74	4632.49	4612.34	4612.34	4551.99	4479.28	4445.26	4443.36	4276.47	4258.98	4041.48
	H	UW	DW-020	DW-100	DEPDT	DQDMR	TEPDT	DW-160	TQDMR	DW-300	DW-900	ZW
TS	Cost	484.07	484.07	472.38	467.08	463.27	463.04	460.16	459.72	451.81	448.14	447.96

Table 5.6: Heuristics' Average DPP Graph Cost Performance Over Different Graphs (REVERSE cost)

	DEPDT	DQDMR	TEPDT	TQDMR	UW	DW-020	DW-100	DW-160	DW-300	ZW	DW-900
Waxman1	2873.07	2714.29	2810.03	2698.84	2865.36	2865.36	2860.39	2782.18	2625.72	2927.60	2522.44
Waxman2	2765.09	2670.43	2727.16	2492.74	2955.12	2870.65	2801.29	2677.73	2536.54	2625.60	2422.16
Waxman3	3047.40	2859.94	2957.87	2779.46	3133.30	3046.04	2915.06	2811.33	2626.31	2914.56	2491.22
Waxman4	2770.25	2644.18	2409.31	2332.58	2585.10	2583.52	2488.20	2436.79	2299.71	2490.75	2228.77
Waxman5	2637.77	2533.50	2670.46	2519.99	2774.67	2727.52	2581.18	2524.40	2440.66	2713.02	2341.41
Waxman6	2645.78	2528.21	2615.62	2526.66	2903.71	2903.71	2847.02	2741.47	2587.52	2984.69	2477.91
Waxman7	2819.81	2699.72	2685.14	2575.11	3006.50	2962.37	2900.89	2786.43	2645.86	2790.33	2524.41
Waxman8	3007.42	2867.03	2840.93	2709.23	3290.05	3218.48	3073.49	2936.38	2741.94	3255.82	2617.03
Waxman9	2580.90	2472.74	2592.29	2398.00	3083.28	3057.27	2707.57	2584.10	2469.74	2693.68	2393.84
Waxman10	2774.31	2605.03	2446.98	2365.98	2667.46	2671.73	2564.44	2487.12	2367.39	2640.44	2284.65
Random	4811.86	4568.55	4461.51	4276.13	4635.88	4635.88	4497.52	4448.40	4291.57	4649.50	4060.45
Locality	4788.74	4551.99	4443.36	4258.98	4612.34	4612.34	4479.28	4445.26	4276.47	4632.49	4041.48
TS	467.08	463.27	463.04	459.72	484.07	484.07	472.38	460.16	451.81	447.96	448.14

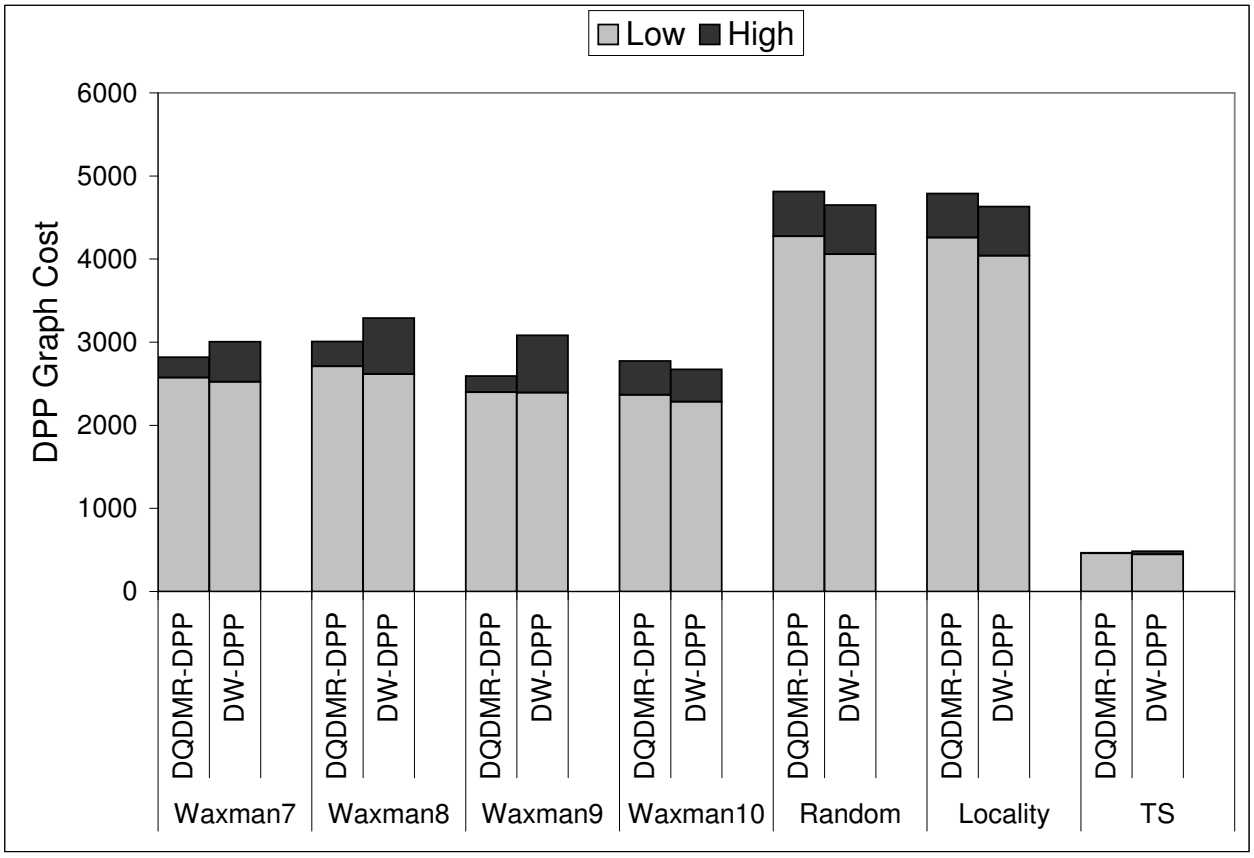

Figure 5.16: Heuristics Performance on Disparate Graphs-II (REVERSE cost)

- DW-DPP ($k = 900$), and DW-DPP ($k = 300$) construct the lowest cost DPP graphs among the different heuristics.
- DW-DPP with increasing k values constructs lower cost DPP graphs.
- DEPDT-DPP generally constructs DPP graphs with higher cost than the other heuristics.
- TQDMR-DPP and TEPDT-DPP consistently construct lower cost DPP graphs than DQDMR-DPP and DEPDT-DPP.

From the disparate graphs' high-low evaluations illustrated in Figure 5.17 and Figure 5.18 it is evident that the low value of DPP graph cost for DW-DPP group heuristics is lower than the low value for DQDMR-DPP group heuristics. And the high value of DPP graph cost is lower in DW-DPP group heuristics than in DQDMR-DPP group heuristics.

5.3 Evaluations on 100 different Waxman Graphs:

In this section the heuristics are all compared on one hundred different 100 node Waxman graphs. The properties of the Waxman graphs generated along with the parameters used during the generation process are shown in Table 5.9. The heuristic runs are simulated on these 100 different Waxman graphs for an extensive comparison of the heuristics' performance. All the heuristics are compared against DQDMR-DPP to measure their relative performance.

Tables 5.10, 5.11, and 5.12 compare the heuristics' performance with respect to DQDMR-DPP for the 100 graphs. The tables from the left to right (along the columns) denote the following:

1. *Heuristic(H):* The heuristic (H) that is being compared to DQDMR-DPP.
2. *Average C_H:* The average cost of multicast trees found by heuristic H ($\frac{Total\ C_H}{Times_H}$).
3. *Average $C_{DQDMR-DPP}$:* The average cost of multicast trees found by DQDMR-DPP ($\frac{Total\ C_{DQDMR-DPP}}{Times_{DQDMR-DPP}}$). As the average $C_{DQDMR-DPP}$ value is the same along all the rows, this column is eliminated and the information is included in the table caption.

Table 5.7: Heuristics' Average DPP Graph Cost Performance Order Over Different Graphs (RANDOM cost)

	H	DEPDT	UW	DW-020	DQDMR	TEPDT	DW-100	ZW	DW-160	TQDMR	DW-300	DW-900
Waxman1	Cost	2314.51	2313.99	2313.99	2242.65	2166.80	2154.31	2115.74	2083.31	2081.05	1990.40	1903.78
	H	DEPDT	DW-020	DQDMR	UW	TEPDT	DW-100	TQDMR	ZW	DW-160	DW-300	DW-900
Waxman2	Cost	2288.11	2218.76	2204.95	2091.63	2073.22	2046.12	2007.79	1974.40	1955.57	1864.94	1781.18
	H	UW	DW-020	DEPDT	DQDMR	DW-100	TEPDT	DW-160	TQDMR	ZW	DW-300	DW-900
Waxman3	Cost	2580.17	2560.35	2498.54	2428.53	2401.16	2289.38	2273.26	2253.47	2220.67	2139.23	2014.03
	H	DEPDT	DQDMR	UW	DW-020	DW-100	TEPDT	TQDMR	DW-160	ZW	DW-300	DW-900
Waxman4	Cost	2665.15	2574.30	2563.68	2563.68	2460.30	2381.37	2320.97	2316.86	2190.10	2154.79	2058.34
	H	DEPDT	UW	DW-020	DQDMR	DW-100	TEPDT	DW-160	TQDMR	ZW	DW-300	DW-900
Waxman5	Cost	2597.81	2574.90	2573.49	2546.84	2501.56	2407.69	2389.14	2357.92	2330.15	2247.05	2141.64
	H	DEPDT	UW	DW-020	DQDMR	TEPDT	DW-100	ZW	DW-160	TQDMR	DW-300	DW-900
Waxman6	Cost	2432.21	2315.66	2315.66	2302.67	2177.97	2162.31	2109.33	2059.17	2059.14	1954.34	1854.94
	H	UW	DW-020	DEPDT	DQDMR	DW-100	TEPDT	DW-160	TQDMR	ZW	DW-300	DW-900
Waxman7	Cost	2452.12	2443.16	2423.74	2359.25	2277.12	2273.43	2184.11	2176.30	2122.93	2072.28	1978.98
	H	DEPDT	DQDMR	DW-020	UW	DW-100	TEPDT	ZW	DW-160	TQDMR	DW-300	DW-900
Waxman8	Cost	2871.09	2723.16	2662.58	2649.48	2508.46	2470.39	2463.43	2412.07	2382.26	2263.06	2146.65
	H	DEPDT	DQDMR	DW-020	UW	TEPDT	DW-100	TQDMR	ZW	DW-160	DW-300	DW-900
Waxman9	Cost	2204.28	2162.91	2144.33	2137.81	2072.96	2060.55	2031.76	2001.41	1997.24	1916.64	1857.02
	H	DEPDT	DQDMR	UW	DW-020	DW-100	TEPDT	DW-160	TQDMR	ZW	DW-300	DW-900
Waxman10	Cost	2885.29	2735.56	2577.95	2566.75	2513.01	2416.12	2399.72	2375.80	2340.06	2269.76	2164.01
	H	DEPDT	DQDMR	UW	DW-020	DW-100	TEPDT	DW-160	ZW	TQDMR	DW-300	DW-900
Random	Cost	4065.02	3897.78	3798.10	3798.10	3757.92	3632.88	3616.11	3588.57	3511.52	3409.26	3224.88
	H	DEPDT	DQDMR	UW	DW-020	TEPDT	DW-100	DW-160	TQDMR	ZW	DW-300	DW-900
Locality	Cost	4174.99	4002.19	3932.72	3932.72	3903.17	3865.32	3763.23	3706.55	3610.16	3506.51	3300.22
	H	UW	DW-020	TEPDT	TQDMR	DEPDT	DQDMR	DW-100	DW-160	DW-300	ZW	DW-900
TS	Cost	506.46	506.46	503.69	501.88	501.75	497.35	489.56	483.84	479.54	477.95	477.57

Table 5.8: Heuristics' Average DPP Graph Cost Performance Over Different Graphs (RANDOM cost)

	DEPDT	DQDMR	TEPDT	TQDMR	UW	DW-020	DW-100	DW-160	DW-300	ZW	DW-900
Waxman1	2314.51	2242.65	2166.80	2081.05	2313.99	2313.99	2154.31	2083.31	1990.40	2115.74	1903.78
Waxman2	2288.11	2204.95	2073.22	2007.79	2091.63	2218.76	2046.12	1955.57	1864.94	1974.40	1781.18
Waxman3	2498.54	2428.53	2289.38	2253.47	2580.17	2560.35	2401.16	2273.26	2139.23	2220.67	2014.03
Waxman4	2665.15	2574.30	2381.37	2320.97	2563.68	2563.68	2460.30	2316.86	2154.79	2190.10	2058.34
Waxman5	2597.81	2546.84	2407.69	2357.92	2574.90	2573.49	2501.56	2389.14	2247.05	2330.15	2141.64
Waxman6	2432.21	2302.67	2177.97	2059.14	2315.66	2315.66	2162.31	2059.17	1954.34	2109.33	1854.94
Waxman7	2423.74	2359.25	2273.43	2176.30	2452.12	2443.16	2277.12	2184.11	2072.28	2122.93	1978.98
Waxman8	2871.09	2723.16	2470.39	2382.26	2649.48	2662.58	2508.46	2412.07	2263.06	2463.43	2146.65
Waxman9	2204.28	2162.91	2072.96	2031.76	2137.81	2144.33	2060.55	1997.24	1916.64	2001.41	1857.02
Waxman10	2885.29	2735.56	2416.12	2375.80	2577.95	2566.75	2513.01	2399.72	2269.76	2340.06	2164.01
Random	4065.02	3897.78	3632.88	3511.52	3798.10	3798.10	3757.92	3616.11	3409.26	3588.57	3224.88
Locality	4174.99	4002.19	3903.17	3706.55	3932.72	3932.72	3865.32	3763.23	3506.51	3610.16	3300.22
TS	501.75	497.35	503.69	501.88	506.46	506.46	489.56	483.84	479.54	477.95	477.57

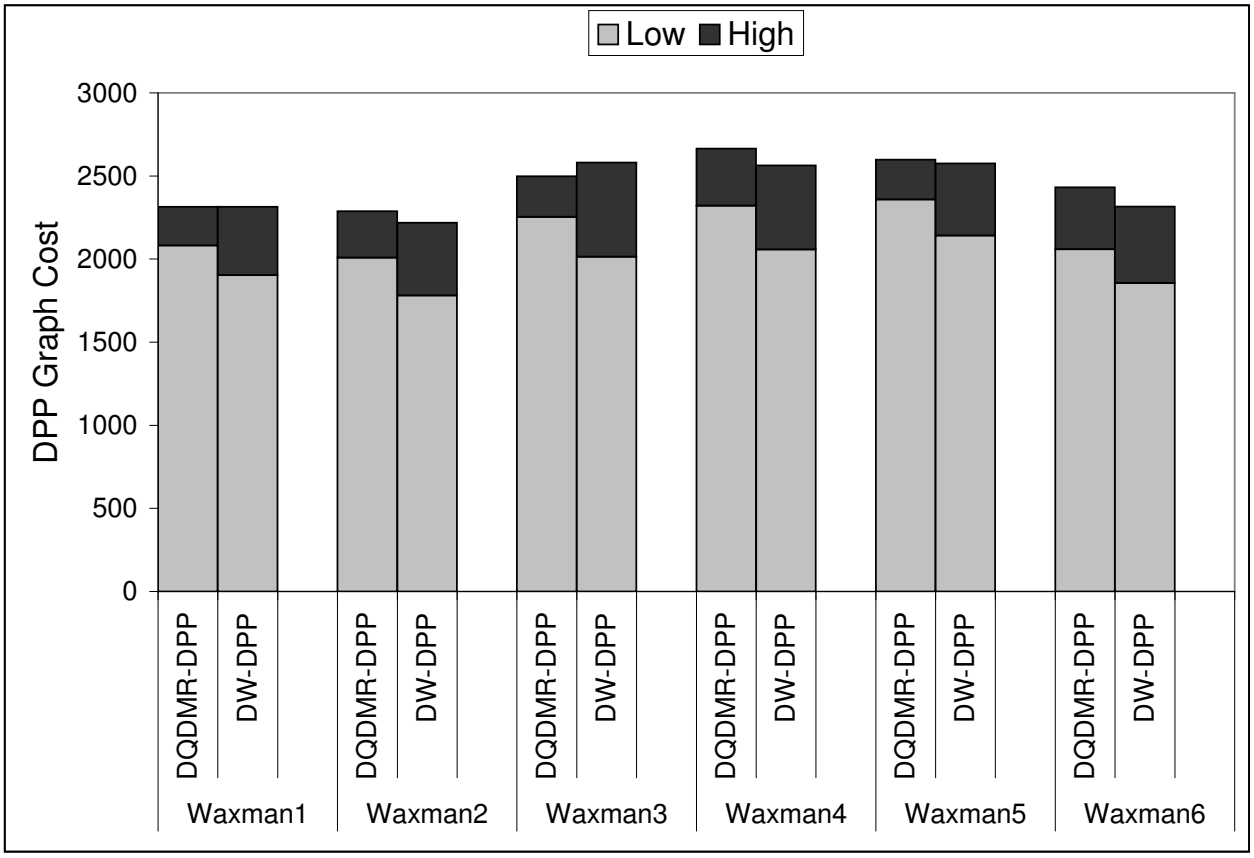

Figure 5.17: Heuristics Performance on Disparate Graphs-I (RANDOM cost)

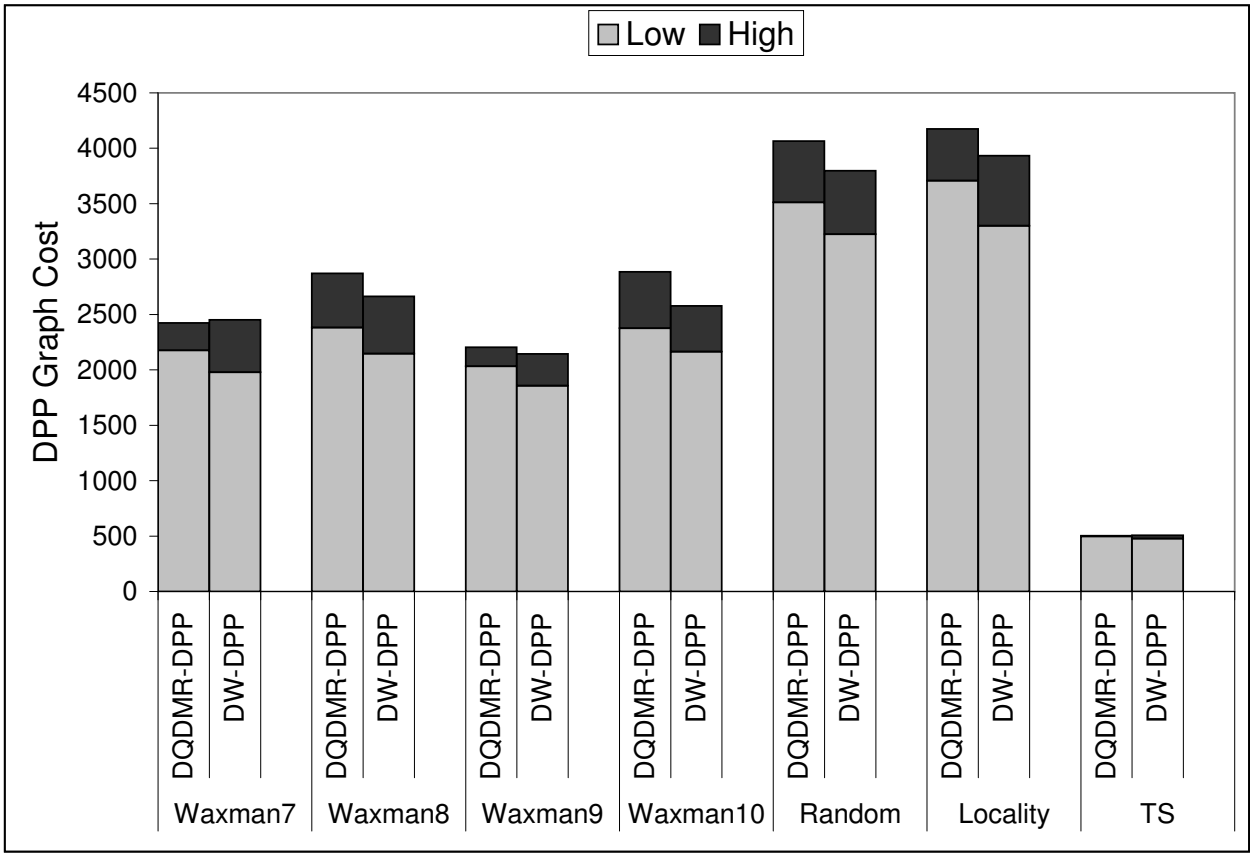

Figure 5.18: Heuristics Performance on Disparate Graphs-II (RANDOM cost)

Table 5.9: Properties of the 100 Waxman Graphs

Waxman	Average Node Degree	Graph Parameters
01 - 05	5.86, 5.66, 5.16, 5.76, 5.68	$\alpha = 0.245, \beta = 0.210$
05 - 10	6.02, 5.76, 5.34, 5.96, 5.84	$\alpha = 0.250, \beta = 0.210$
10 - 15	6.14, 5.94, 5.50, 6.12, 6.10	$\alpha = 0.250, \beta = 0.215$
15 - 20	6.26, 6.04, 5.58, 6.30, 6.32	$\alpha = 0.255, \beta = 0.215$
20 - 25	6.46, 6.22, 5.74, 6.54, 6.50	$\alpha = 0.255, \beta = 0.220$
25 - 30	6.58, 6.28, 5.86, 6.68, 6.58	$\alpha = 0.260, \beta = 0.220$
30 - 35	6.82, 6.44, 6.04, 6.84, 6.70	$\alpha = 0.260, \beta = 0.225$
35 - 40	6.94, 6.50, 6.20 , 6.96, 6.86	$\alpha = 0.265, \beta = 0.225$
40 - 45	7.04, 6.64, 6.34, 7.22, 7.00	$\alpha = 0.265, \beta = 0.230$
45 - 50	7.20, 6.74, 6.42, 7.30, 7.12	$\alpha = 0.270, \beta = 0.230$
50 - 55	7.36, 6.82, 6.66, 7.50, 7.22	$\alpha = 0.270, \beta = 0.235$
55 - 60	7.52, 7.00, 6.80, 7.62, 7.32	$\alpha = 0.275, \beta = 0.235$
60 - 65	7.70, 7.12, 6.96, 7.76, 7.48	$\alpha = 0.275, \beta = 0.240$
65 - 70	7.74, 7.20, 7.12, 7.84, 7.76	$\alpha = 0.280, \beta = 0.240$
70 - 75	7.88, 7.46, 7.22, 7.96, 7.84	$\alpha = 0.280, \beta = 0.245$
75 - 80	8.02, 7.64, 7.44, 8.16, 8.06	$\alpha = 0.285, \beta = 0.245$
80 - 85	8.14, 7.80, 7.72, 8.34, 8.24	$\alpha = 0.285, \beta = 0.250$
85 - 90	8.24, 7.90, 7.84, 8.52, 8.36	$\alpha = 0.290, \beta = 0.250$
90 - 95	8.46, 8.06, 8.02, 8.64, 8.50	$\alpha = 0.290, \beta = 0.255$
95 - 100	8.60, 8.08, 8.06, 8.86, 8.66	$\alpha = 0.295, \beta = 0.255$

4. $\frac{C_H}{C_{DQDMR-DPP}}$: The ratio of the average cost of trees found by H over that found by DQDMR-DPP over all instances $\frac{\text{Average } C_H}{\text{Average } C_{DQDMR-DPP}}$.

The heuristics are arranged such that the highest average tree cost heuristics are listed at the top and the lowest average tree cost heuristics are listed at the bottom.

5.3.1 EQUAL cost scenario

Table 5.10 shows the relative difference in the performance of the heuristics over 100 different Waxman graphs in the EQUAL cost scenario with respect to DQDMR-DPP. Listed below are the conclusions from the data shown in the table:

- DW-DPP (at $k = 20$), DEPDT-DPP, UW-DPP and DQDMR-DPP on an average construct DPP graphs with the highest cost. Note that the ratio $\frac{C_H}{C_{DQDMR-DPP}}$ is 1 for DQDMR-DPP.
- DW-DPP (at $k = 900$), ZW-DPP on an average construct DPP graphs with lowest cost.
- The heuristics have the following performance order from worst to best in terms of DPP graph cost over the 100 different graphs (DW-DPP-020, DEPDT-DPP, UW-DPP, DQDMR-DPP, DW-DPP-100, TEPDT-DPP, TQDMR-DPP, DW-DPP-160, DW-DPP-300, ZW-DPP, and DW-DPP-900).
- DW-DPP at ($k \geq 160$) constructs on an average lower cost DPP graphs than the best online variants of QDMR (TQDMR-DPP and TEPDT-DPP).
- The percentage difference in the performance of the best heuristic (DW-DPP at $k = 900$) and the worst heuristic (DW-DPP at $k = 20$) is 39.33%. DW-DPP with increasing k constructs DPP graphs with lower cost.
- The percentage difference in the performance of DW-DPP at $k = 900$ and DQDMR-DPP is 36.26%.

5.3.2 *REVERSE cost scenario*

Table 5.11 shows the relative difference in the performance of the heuristics over 100 different Waxman graphs in the REVERSE cost scenario with respect to DQDMR-DPP. Listed below are the conclusions from the data shown in the table:

- DW-DPP (at $k = 900$) constructs DPP graphs with lower cost than the other heuristics.
- UW-DPP and DW-DPP (at $k = 20$) on an average construct DPP graphs with higher cost than other heuristics.
- The heuristics have the following performance order from worst to best in terms of tree cost over the 100 different graphs (UW-DPP, DW-DPP-020, ZW-DPP, DW-DPP-100, DEPDT-DPP, DW-DPP-160, DQDMR-DPP, TEPDT-DPP, DW-DPP-300, TQDMR-DPP, DW-DPP-900).
- DQDMR-DPP constructs DPP graphs on an average with lower cost than UW-DPP, DW-DPP ($k = 20$), ZW-DPP, DW-DPP ($k = 100$), respectively.
- The percentage difference in the performance of the best heuristic (DW-DPP at $k = 900$) and the worst heuristic (UW-DPP) is 20.65%.
- The percentage difference in the performance of DW-DPP at $k = 900$ and DQDMR-DPP is 5.48%.
- DW-DPP with increasing k constructs DPP graphs with lower cost.

5.3.3 *RANDOM cost scenario*

Table 5.12 shows the relative difference in the performance of the heuristics over 100 different Waxman graphs in the RANDOM cost scenario with respect to DQDMR-DPP. Listed below are the conclusions from the data shown in the table:

Table 5.10: Heuristics' Cost Comparison wrt DQDMR-DPP over 100 graphs (EQUAL cost), Average $C_{DQDMR-DPP}$=2617.564

Heuristic(H)	Average C_H	$\frac{C_H}{C_{DQDMR-DPP}}$
DW-DPP-020	2676.584	1.02255
DEPDT-DPP	2671.639	1.02066
UW-DPP	2657.625	1.01530
DW-DPP-100	2354.915	0.89966
TEPDT-DPP	2315.608	0.88464
TQDMR-DPP	2219.666	0.84799
DW-DPP-160	2157.021	0.82406
DW-DPP-300	2004.411	0.76575
ZW-DPP	1960.684	0.74905
DW-DPP-900	1920.957	0.73387

- DEPDT-DPP, DW-DPP (at $k = 20$), UW-DPP, and DQDMR-DPP on an average construct DPP graphs with the highest cost.
- DW-DPP (at $k = 900$) on an average constructs the lowest cost DPP graphs.
- The heuristics have the following performance order from worst to best in terms of tree cost over the 100 different graphs DEPDT-DPP, DW-DPP-020, UW-DPP, DQDMR-DPP, DW-DPP-100, TEPDT-DPP, DW-DPP-160, TQDMR-DPP, DW-DPP-300, ZW-DPP, DW-DPP-900
- The percentage difference in the performance of the best heuristic (DW-DPP $k = 900$) and the worst heuristic (DEPDT-DPP) is 24.14%.
- The percentage difference in the performance of DW-DPP at $k = 900$ and DQDMR-DPP is 19.83%
- DW-DPP with increasing k constructs DPP graphs with lower cost.

Table 5.11: Heuristics' Cost Comparison wrt DQDMR-DPP over 100 graphs (REVERSE cost), Average $C_{DQDMR-DPP}$=2625.886

Heuristic(H)	Average C_H	$\frac{C_H}{C_{DQDMR-DPP}}$
UW-DPP	3003.282	1.14372
DW-DPP-020	2967.209	1.12998
ZW-DPP	2916.534	1.11069
DW-DPP-100	2845.639	1.08369
DEPDT-DPP	2758.636	1.05055
DW-DPP-160	2735.721	1.04183
TEPDT-DPP	2615.255	0.99595
DW-DPP-300	2596.797	0.98892
TQDMR-DPP	2503.200	0.95328
DW-DPP-900	2489.344	0.94800

Table 5.12: Heuristics' Cost Comparison wrt DQDMR-DPP over 100 graphs (RANDOM cost), Average $C_{DQDMR-DPP}$=2390.514

Heuristic(H)	Average C_H	$\frac{C_H}{C_{DQDMR-DPP}}$
DEPDT-DPP	2476.591	1.03601
DW-DPP-020	2448.107	1.02409
UW-DPP	2447.766	1.02395
DW-DPP-100	2310.680	0.96660
TEPDT-DPP	2246.365	0.93970
ZW-DPP	2218.875	0.92820
DW-DPP-160	2208.305	0.92378
TQDMR-DPP	2168.806	0.90726
DW-DPP-300	2084.627	0.87204
DW-DPP-900	1994.960	0.83453

5.4 Comparison to the optimal solution:

The heuristics' are all compared to the optimal solution on a small graph. The minimum cost delay constrained disjoint path multicast routing problem can be solved on small graphs through the use of exhaustive enumeration. A 10 node 16 edge graph is used in the comparison process. The exploration effort in exhaustively enumerating the 65536 edge subsets is reduced by using a filtration tecnhique:

- Determine whether the edge subset forms a connected component.
- Determine whether the connected component has the source and the all the destination nodes.
- Determine whether both the paths in the path pair to each destination along the connected component meet the delay bound of the application.

From among the connected components that meet above listed requirements select one which has the lowest cost. This disjoint path pair graph represents the lowest achievable cost (AC) for a delay-constrained minimum cost disjoint path multicast routing problem.

Figure 5.19 shows the cumulative distribution of the *relative DPP graph cost* obtained by the heuristics, compared to the optimal DPP graph constructed for 100 different multicast destination sets. Relative DPP graph cost of a heuristic is the ratio of cost of the DPP graph constructed by the heuristic divided by AC.

This comparison to the optimal solution confirms the fact that DW-DPP (at high $k \geq 30$) and ZW-DPP construct DPP graphs with lower cost than the other heuristics. DW-DPP (at $k = 30$ and $k = 50$) has a worst relative tree cost of 2.0 which is better than the other online heuristics. UW-DPP has the worst relative DPP graph cost performance. The relative DPP Graph cost performance of QDMR based variants is close to that of DW-DPP (at $k = 20$). The relative DPP graph cost performance of DW-DPP improves with increasing k.

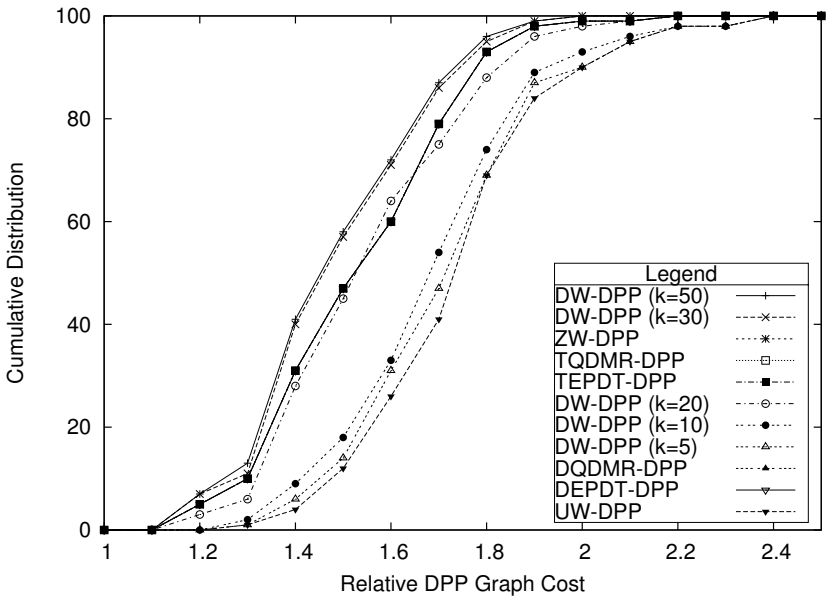

Figure 5.19: Cumulation distribution of the relative cost of the heuristics ($|N| = 10$, $|E| = 16$, $m = 6$, $\Delta = 50$)

5.5 DW-DPP behavior at varying k

In order to predict an appropriate k value for DW-DPP, its behavior at various k values is observed both at different delay bound values on a single Waxman graph and at a fixed delay bound value on four different Waxman graphs (with varying node degree).

Figure 5.20 plots the change in DW-DPP performance (with respect to average DPP graph cost) for varying k at different delay bound values $\Delta = \{140, 160, 300, 600, 900\}$ (representing a wide range of the delay requirements of the application) on a single 100 node Waxman graph (average node degree=5.86). The value for k is changed from 20 to 3000 in increments of 20. From the figure it is clear that the average DPP graph cost performance improves with increasing k at all delay bound values.

At low k values as DW-DPP is more focussed on the delay performance, the lowest-effective-cost path pairs computed at this k may fail to meet the delay requirement of the application, requiring the adding of the more costly least-delay path pairs and hence resulting in higher cost DPP graphs. However at higher k DW-DPP is more focussed on the cost performance and constructs

Table 5.13: DW-DPP graph cost performance at different k

k	Average DPP Graph Cost	Difference wrt $k = 3000$
20	2452.90	24.30%
100	2274.16	15.24%
160	2143.26	8.61%
300	2039.72	3.37%
600	1990.92	0.89%
900	1986.18	0.65%
1800	1976.70	0.17%
3000	1973.31	0%

lower cost DPP graphs. This is because, the weight coefficients assigned to the edges decrease with increasing k resulting in higher likelihood of them being re-used reducing the DPP graph cost.

We observe the same DW-DPP behavior with respect to k when run on four different 100 node Waxman graphs (average node degrees=5.86, 6.86, 7.88, 8.86) as shown in Figure 5.21 (k value is changed from 20 to 3000 in increments of 20). From the figure it is clear that the cost of the graphs constructed by DW-DPP generally decreases with increasing k.

The DPP graph cost performance at different k values at ($\Delta = 300$) when run on the 100 node Waxman graph (average node degree=5.86) is listed in Table 5.13. The first and second columns in the table list the k value and the cost of DPP graph constructed by DW-DPP at that k. The third column lists the percentage difference in DPP graph cost for the graphs constructed by DW-DPP at that k (along the row) and DW-DPP at $k = 3000$. The DPP graph cost performance of DW-DPP (with respect to its performance at $k = 3000$) improves from being 24.30% higher (at $k = 20$), to being 8.61% higher (at $k = 160$), to being 0.65% higher (at $k = 900$). Therefore DW-DPP at any $k \geq 900$ constructs low cost DPP graphs and therefore any $k \geq 900$ is an appropriate k value for DW-DPP in this environment.

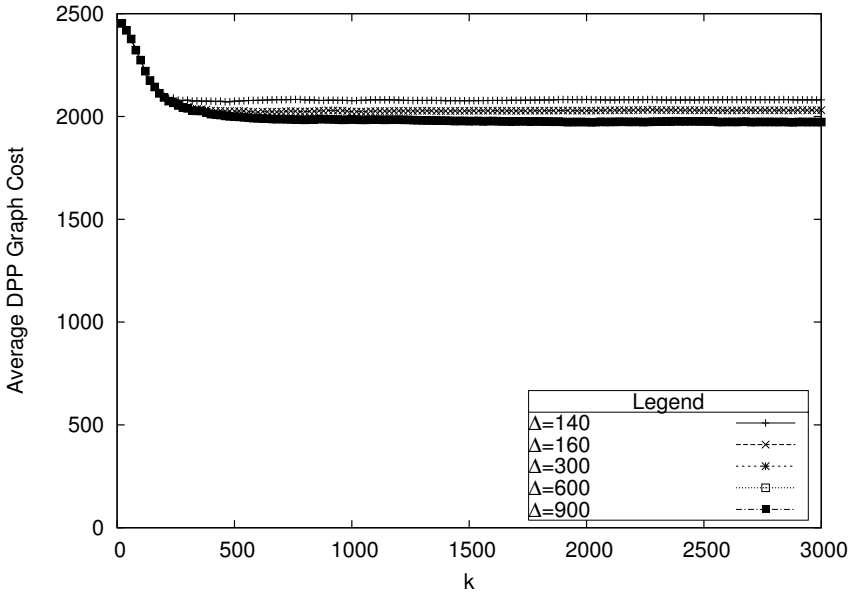

Figure 5.20: Average DPP graph cost versus k at $m = 40$ on a Waxman graph: 100 Nodes, Degree=5.86, $\alpha = 0.245$, $\beta = 0.210$

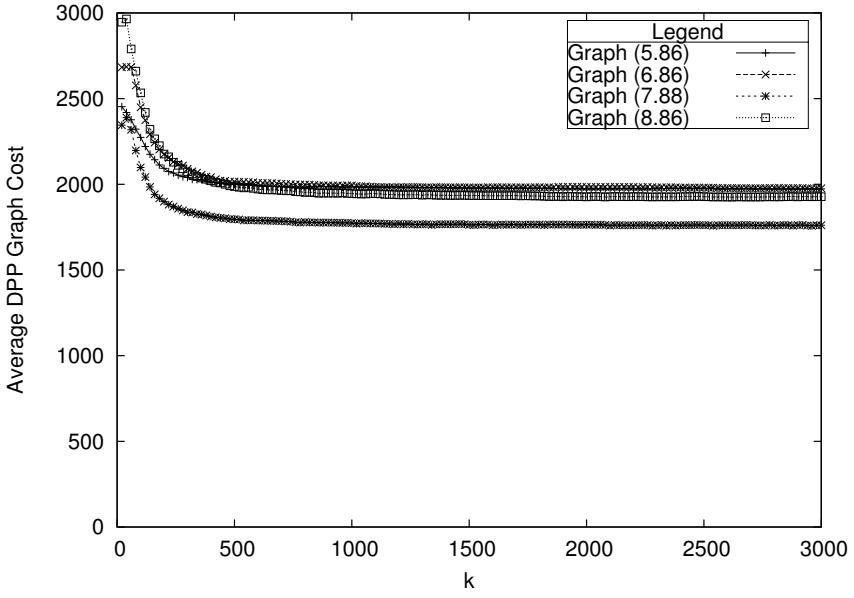

Figure 5.21: Average DPP graph cost versus k at $m = 40$, and $\Delta = 300$ on four Waxman graphs: 100 Nodes, Degrees={5.86, 6.86, 7.88, 8.86}

5.6 Summary

This chapter presents the evaluation of all the online, survivable, low-cost, delay-constrained multicast routing heuristics considered in this work. The evaluations are conducted in a structured manner and result in the following conclusions for all the three cost scenarios:

5.6.1 Evaluation of the heuristics on 4 Waxman graphs

- DQDMR-DPP and DEPDT-DPP generally construct higher cost multicast graphs than all other heuristics.
- TQDMR-DPP and TEPDT-DPP generally construct lower cost multicast graphs than DQDMR-DPP and DEPDT-DPP.
- DW-DPP (at $k = 900$) constructs lower cost multicast graphs than TQDMR-DPP.
- ZW-DPP constructs higher cost DPP graphs at lower delay bound values and higher cost DPP graphs at higher delay bound values.
- DW-DPP (at $k = 900$), and ZW-DPP generally construct lower cost multicast graphs than all other heuristics.

5.6.2 Evaluation on disparate graphs

- DW-DPP (at $k = 900$) on average constructs lower cost multicast graphs than all other heuiristics.
- TQDMR-DPP and TEPDT-DPP construct lower cost multicast graphs than DQDMR-DPP and DEPDT-DPP.
- DW-DPP with increasing k constructs lower cost multicast graphs.

5.6.3 *Evaluations on 100 different Waxman graphs*

- DW-DPP (at $k = 900$) on an average constructs lower cost multicast graphs than all other online heuristics.
- DW-DPP (at $k = 900$) constructs multicast graphs with 16%, 1%, and 9% lower cost than the best node-priority based heuristic TQDMR-DPP for EQUAL, REVERSE, and RANDOM cost scenarios respectively.
- TQDMR-DPP constructs multicast graphs with 18%, 5%, and 10% lower cost than DQDMR-DPP for EQUAL, REVERSE, and RANDOM cost scenarios respectively.

5.6.4 *Comparison to the optimal solution*

- DW-DPP (at $k = 50$ and $k = 30$) constructs multicast graphs with a worst relative graph cost of 2.0 (compared to the optimal solution) which is better than the other online heuristics.

5.6.5 *DW-DPP behavior at varying k*

- The DPP graph cost performance of DW-DPP improves with increasing k.
- DW-DPP (at $k \geq 900$) constructs multicast graphs with lower cost than all other heuristics.
- Any $k \geq 900$ is an appropriate k for DW-DPP when it is run on graphs resembling those used in the evaluations in this work.

CHAPTER SIX

CONCLUSION

This dissertation proposes and evaluates heuristics for the construction of both survivable and non-survivable low-cost, delay-constrained multicast graphs. The new heuristics were motivated from the need for such heuristics in critical infrastructure frameworks such as GridStat. The heuristics and their evaluation in a structured manner will not only help in applications such as GridStat but will also be a useful resource for future researchers.

The dynamic weight heuristic (DWH) is a new heuristic proposed in this work which produces low-cost, delay-constrained multicast trees. DWH assigns dynamic weights to edges in the tree while computing paths. These edge weights are then used as indicators of the relative merit of including the edge in the paths. Edge weights are computed based on the closeness of the delay (from the source to the end of the edge) to the parameter k. If in the first phase the path found using delay-influenced edge-costs fails to meet the delay bound of the application, then a second phase merges the least-delay path to that destination to the multicast tree (if the delay along the least-delay path meets the delay bound). Therefore, DWH meets the twin goals of having a low-cost and a delay-constrained multicast tree.

Two variants of DWH, the least-cost-path variant UWH, and the least total cost variant ZWH are also considered in the evaluation process. These heuristics are then compared to QDMR and its online variants, DQDMR and DEPDT (which assign variable priority to the destination nodes in the tree). Two improved heuristics based on DQDMR and DEPDT are also proposed and evaluated. They are the tree node priority based variants of QDMR, TQDMR and TEPDT.

The DWH heuristic developed for low-cost, delay-constrained multicast routing is modified by making it compute shared disjoint path pairs instead of individual paths to the destination nodes. The modified heuristic is called the Dynamic Weight Disjoint Path Pairs (DW-DPP) heuristic. The path pairs are computed using Bhandari's vertex splitting algorithm. The online heuristics

used in the evaluation of non-survivable multicast routing are modified, like before, to construct shared disjoint path pairs to each destination node. The modified heuristics are UW-DPP, ZW-DPP, DQDMR-DPP, DEPDT-DPP, TQDMR-DPP, and TEPDT-DPP.

Both the survivable and non-survivable heuristics are evaluated in a structured manner to help appreciate the differences between the heuristics and see how the performance varies with varying characteristics of the graphs used in the evaluation:

1. The heuristics are compared on four different 100 node Waxman graphs with a difference of about one in the node degree between adjacent graphs. This enables the evaluation of the heuristics on graphs of varying node degree but of the same type. The evaluations show that DWH and DW-DPP (at $k = 900$) construct lower cost multicast graphs than TQDMR and TQDMR-DPP (the best node-priority-based heuristic) respectively. Further TQDMR and TQDMR-DPP construct lower cost multicast graphs than DQDMR and DQDMR-DPP respectively.

2. The heuristics are evaluated on Waxman, Pure Random, Locality, and Transit-Stub graphs. The heuristic's performance varies with the type of graph used in the evaluation process as has been demonstrated in the evaluations. The evaluations show that DWH and DW-DPP with increasing k construct lower cost multicast graphs.

3. The heuristics are also evaluated on 100 different Waxman graphs. These evaluations cover a range of graph characteristics and are useful in establishing the average performance of the heuristics over a large number of graphs. The evaluations show that DWH (at $k = 900$) constructs multicast trees with 10%, 3%, and 5% lower cost than TQDMR and DW-DPP (at $k = 900$) constructs multicast trees with 16%, 1%, and 9% lower cost than TQDMR-DPP for EQUAL, REVERSE, and RANDOM cost scenarios respectively.

4. The heuristics are also compared to the optimal solution to assess their relative performance

with respect to the lowest achievable solution. The evaluations show that DWH and DW-DPP (at $k = 50$) construct multicast trees with a worst relative tree cost of 1.4 and 2.0 respectively which are better then the other online heuristics.

Also, the heuristics produce costs that are within 28% and 40% of each other for the non-survivable and survivable cases respectively. For a system designer this gives an estimate of the benefit that the more sophisticated multicast heuristics may provide in reducing the cost of the multicast graphs (trees).

Finally, for the evaluation environments used in this research work and as shown in Section 5.5, DW-DPP improves its DPP graph cost performance with increasing k and at ($k \geq 900$) generally outperforms other heuristics and is the ideal heuristic for such environments.

6.1 Future Work

The future work in this research will include the implementation of the DW-DPP heuristic on the GridStat publish-subscribe middleware framework. DW-DPP (at high k values) as shown in Chapter 5 performs better than any other heuristic in terms of its DPP graph cost performance. Also, the DPP graph cost performance of DW-DPP improves with increasing k as shown in Section 5.5.

Like in Section 5.5 an appropriate k value for DW-DPP for use in GridStat can be determined by running a few experiments. In each experiment DW-DPP behavior at different k values (for example in Section 5.5 the k value is varied from 20 to 3000 in increments of 20) for a specific delay bound value when run on the GridStat target graph environment is plotted to ascertain the most appropriate k value. The experiments are run at different delay bound values representing the different hard-to-satisfy and easy-to-satisfy delay bound cases of the GridStat environment. These experiments will help the GridStat researcher in selecting an appropriate k value for DW-DPP for use in GridStat.

Another important point which a GridStat researcher has to contend with during the design

phase is the selection of appropriate cost values for the different edges in the graph. Presently we consider three different delay-cost relationships (EQUAL, REVERSE, and RANDOM) for assigning cost values to edges based on the delays on the edges. Of these the RANDOM scenario (there is no relationship between the delay and cost of an edge and the cost values assigned to edges are delays of different graph edges selected at random) represents the most practical delay-cost relationship.

However cost of an edge is not currently coupled with the resources available on the edge. An important question which the GridStat researcher will have to answer is whether or not the cost of an edge in the graph should be coupled to the resources available on the edge and whether this leads to a more accurate reflection of the actual cost of using the edge in multicast routing.

Also as seen in the non-survivable case (Chapter 4) offline QDMR heuristic's tree cost performance is better still than the online DWH heuristic. If the preliminary multicast destination set (representing the potential preliminary subscribers) in GridStat is known ahead of time and is relatively stable, that is only a few additional subscribers are expected to join the multicast session after the start of the session, then there is potentially some benefit in using a hybrid offline-online heuristic for constructing the multicast graph. This heuristic first computes an offline multicast graph for the known destination set (known subscribers) and incrementally expands (online part of the heuristic) the constructed graph by adding a path pair to each new subscriber that joins the multicast session.

However there are no offline survivable delay-constrained low-cost multicast routing heuristics existing in the literature and the future work will involve development of such heuristics. Also a hybrid offline-online heuristic when developed will make it possible for the GridStat researcher to compare and evaluate the benefits of using such a hybrid heuristic over a purely online heuristic such as DW-DPP in GridStat.

BIBLIOGRAPHY

[1] M. Garey, and D. Johnson, "Computers and Intractability: A Guide to the theory of NP-completeness," *New York: W. H. Freeman and Co.,* 1979.

[2] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica,* vol. 15, pp. 141-145, 1981.

[3] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Math. Japonica,* vol. 24, pp. 573-577, 1980.

[4] G. Robins and A. Zelikovsky, "Improved Steiner Tree Approximation in Graphs," *Proceedings of the eleventh annual ACM/SIAM Symposium on Discrete Algorithms (SODA 2000),* pp. 770-779, Jan 2000.

[5] C.H. Hauser, D.E. Bakken, and A. Bose, "A failure to communicate: next generation communication requirements, technologies, and architecture for the electric power grid" *IEEE Power and Energy Magazine,* vol. 3, no. 2, pp. 47-55, Mar.-Apr. 2005.

[6] C.H. Hauser, D.E. Bakken, I. Dionysiou, K.H. Gjermundrd, V.S. Irava, and A. Bose, "Security, trust, and QoS in next-generation control and communication for large power systems", *Proceedings of the Workshop on Complex Network and Infrastructure Protection (CNIP06),* Rome, 28-29 March, 2006.

[7] K.H. Gjermundrod, "Flexible QoS-Managed Status Dissemination Middleware Framework for the Electric Power Grid," *PhD thesis, Washington State University,* Pullman, Washington, Aug. 2006. Available from http://griffin.wsu.edu.

[8] "GridStat," *www.gridstat.net,* Washington State University, 2006.

[9] L. Guo and I. Matta, "QDMR: An efficient QoS Dependent Multicast Routing Algorithm," *Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium (RTAS '99),* pp 213, 1999.

[10] A. Fei, J. Cui, M. Gerla, and D. Cavendish, "A dual-tree scheme for fault-tolerant multicast," *Proceedings of IEEE International Conference on Communications, 2001,* vol. 3, pp.690-694, Jun. 2001.

[11] M. Aissa, and A. B. Mnaouer, "A new delay constrained algorithm for multicast tree construction," *International Journal of Communication Systems*, vol. 17, no. 10, pp. 985-1000, Dec. 2004.

[12] N. K. Singhal, L. H. Sahasrabuddhe, and B. Mukherjee, "Provisioning of Survivable Multicast Sessions Against Single Link Failures in Optical WDM Mesh Networks," *Journal of Lightwave Technology,* vol. 21, no. 11, pp. 2587-2594, Nov. 2003.

[13] P. Erdös, and A. Rényi, "On Random Graphs I," *Publicationes Mathematicae Debrecen,* vol. 6, pp. 290-297, 1959.

[14] B. M. Waxman, "Routing of Multipoint Connections," *IEEE Journal on Selected Areas in Communications,* vol. 6, no. 9, pp. 1617-1622, Dec. 1988.

[15] G. N. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," *IEEE Journal on Selected Areas in Communication,* vol. 15, no. 3, pp. 346-356, Apr 1997.

[16] A. Shaikh and K. Shin, "Destination-driven routing for low-cost multicast," *IEEE Journal on Selected Areas in Communication,* vol. 15, no. 3, pp. 373-381, Apr. 1997.

[17] J. W. Suurballe, "Disjoint paths in a network," *Networks,* vol. 4, pp. 125-145, 1974.

[18] R. Bhandari, "Survivable Networks: Algorithms for Diverse Routing," *Kluwer Academic Publishers,* 1999.

[19] L. R. Ford, Jr. and D. R. Fulkerson, "Flows in Networks," *Princeton University Press,* 1962.

[20] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A Quantitative Comparision of Graph-based Models for Internet Topology," *IEEE/ACM Transactions on Networking,* vol. 5, no. 6, pp. 770-783, 1997.

[21] M. Doar, "A Better Model for Generating Test Networks," *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM),* pp. 86-93, Nov. 1996.

[22] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based vs. Structural," *ACM SIGCOMM,* vol. 32, no. 4, pp. 147-159, 2002.

[23] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-Law Relationships of the Internet Topology," *ACM SIGCOMM,* vol. 29, no. 4, pp. 251-262, 1999.

[24] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," *Proceedings of the 32nd Annual Symposium on Theory of Computing,* 2000.

[25] R. Albert, and A.L. Barabási, "Topology of evolving networks: local events and universality," *Physical Review Letters,* vol. 85, no. 24, pp. 5234-5237, Dec. 2000.

[26] R. Albert, and A.L. Barabási, "Emergence of scaling in random networks," *Science,* vol. 286, pp. 509-512, 1999.

[27] R. Govindan, and H. Tangmunarunkit, "Heuristics for Internet Map Discovery," *Proceedings of the IEEE INFOCOM,* vol. 3, pp. 1371-1380, Mar. 2000.

[28] G. Phillips, S. Shenker, and H. Tangmunarunkit, "Scaling of Multicast Trees: Comments on the Chuang-Sirbu Scaling Law," *Proceedings of the ACM SIGCOMM Computer Communication Review,* vol. 29, no. 4, pp. 41-51, 1999.

[29] G. Karypis, and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing,* vol. 20, no. 1, pp. 359-392, Aug. 1998.

[30] T. Hu, "Optimum Communication Spanning Trees," *SIAM Journal of Computing,* vol. 3, no. 3, pp. 188-195, 1974.

[31] JH. Cui, M. Faloutsos, D. Maggiorini, M. Gerla, and K. Boussetta, "Measuring and modelling the group membership in the Internet, " *Proceedings of the 3rd ACM SIGCOMM conferene on Intermet measurement,* pp. 65-77, 2003.

[32] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," *Proceedings of SIGCOMM'00,* pp. 97-110, 2000.

[33] D. Dolev, O. Mokryn, and Y. Shavitt, "On Multicast Trees: Structure and Size Estimation," *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM' 03),* vol. 2, pp. 1011-1021, 2003.

[34] W. Cheswick, J. Nonnenmacher, C. Sahinalp, R. Sinha, and K. Varadhan, "Modeling internet topology," *Lucent Technologies, Tech. Rep. Technical Memorandum 113410-991116-18TM,* 1999.

[35] P. Rajvaidya, and K. Almeroth, "Analysis of Routing Characteristics in the Multicast Infrastructure," *Proceedings of IEEE INFOCOM,* 2003.

[36] J. D. Bansemer, and M. Y. Eltoweissy, "On Performance Metrics for IP Multicast Routing,"

Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'00), pp. 282-291, 2000.

[37] J. Huang, X. Du, Z. Yang, and W. Cheng, "Available Bandwidth-Based Real-Time Multicast Routing Distributed Algorithm," *International Conference on Computer Networks and Mobile Computing (ICCNMC'03),* 2003.

[38] Sajama, and Z. J. Haas, "Independent-Tree Ad hoc Multicast Routing (ITAMAR)," *Mobile Networks and Applications,* vol. 8, no. 5, pp. 551-566, Oct. 2003.

[39] V.S. Irava, and C. Hauser, "Survivable low-cost low-delay multicast trees," *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM),* Nov. 2005.

[40] F. Bauer, and A. Varma, "ARIES: A rearrangeable inexpensive edge-based on-line Steiner algorithm," *IEEE Journal on Selected Areas in Communication,* vol. 15, pp. 382-397, Apr. 1997.