Proceedings of the IASTED International Conference
Parallel and Distributed Computing and Systems (PDCS 2011)
December 14 - 16, 2011 Dallas, USA

SP-MULTICAST TREES FOR GRIDS

Teofilo F. Gonzalez
Department of Computer Science
University of California, Santa Barbara, CA, 93106, USA
e-mail: teo@cs.ucsb.edu

ABSTRACT
We study the problem of constructing a shortest path multicast (sp-multicast) tree for transmitting a message from a node, s, to a set of destinations, D, in $2d$-grid networks. We show that the performance analysis of the previous 2-approximation algorithms is best possible. Given any half-integral optimal solution, we present a global rounding scheme that generates solutions that are within 1.5 times the optimal solution value. Our algorithm is different from previous ones in the sense that it is not based on simple heuristics or brute force approaches, it is based on a global rounding strategy. Our analysis uses a sharper lower bound.

KEY WORDS
Approximation algorithms, $2d$-grid network, sp-multicast trees, multicast, rectilinear Steiner tree arborescence.

1 Introduction

Multicast is a basic operation used by systems and algorithms to disseminate information from a source node (s) to a set of destination nodes (D) through a given communication networks. The simplest way to implement a multicast operation is by sending individual messages from node s to each of the destination nodes. However, when the messages are long, there are many destinations, or the network destinations are far from the source node, we need to implement the multicast operation efficiently so as to minimize the total number of communication links transporting the message. For an efficient implementation, one constructs a communication tree, rooted at node s, consisting of network links and including all the destination nodes. We say that the multicast tree is an *sp-multicast* tree if, every path from the source node s to every destination $d \in D$ in the multicast tree has a number of links which is equal to the number of links in a shortest path from s to d in the (whole) communication network. Our problem is to construct an sp-multicast tree with the fewest number of links. This objective function minimizes the network resources (links) being used to transmit the message, while delivering the message to all its destinations via a shortest possible route. In other words, it is a dual objective function where we optimize the length of every communication path, while using the fewest total number of communication links. In this paper, we restrict our attention to the $2d$-grid communication network. Our objective is to find good approximation algorithms for this problem, which could be extended to other architectures, like the n-cube, n-Chord and binomial networks. There are no known constant ratio approximation algorithms for these more general problems. For $c > 1$ we say that an algorithm is a *c-approximation* algorithm for a problem P, if for every instance, I, of P, the algorithm generates a solution with objective function value that is at most c times the objective function value of an optimal solution. We also say that the *approximation ratio* of the algorithm is c.

Assign to each node x in a $2d$-grid a value equal to the number of links in a shortest path from x to s. We call this value b_x for node x. One can easily show that every link joins two nodes that differ in b values by exactly one. Therefore, we can direct every link from the node with larger value to the one with smaller value. The $2d$-grid is now a directed graph and our sp-multicasting tree problem corresponds to finding an arborescence, rooted at s, with the fewest number of links. This problem has been referred to as the *Rectilinear Steiner Arborescence (RSA)* problem.

The rectilinear Steiner arborescence problem was initially studied by Ladeira de Matos [1] back in 1979. His Ph.D. Dissertation includes dynamic programming algorithms to solve this problem. These algorithms have exponential time complexity, and thus impractical even for small problem instances. Nastansky, Selkow and Stewart [2] developed integer programming formulations for the RSA problem when generalized to d-dimensions. But since integer programing is an NP-hard problem, the fastest known algorithms to solve ILP problems take exponential time. Trubin [3] claimed to have developed a polynomial time algorithm to solve the 2d-grid RSA problem. However, the algorithm was shown to generate non-optimal solutions by Rao, Sadayappan, Hwang and Shor [4]. Trubin's algorithm is rather complicated to follow. Initially an integer programming problem is formulated. Then its dual relaxation problem is solved via linear programming. The claim is that every instance of the dual relaxation problem has at least one optimal solution that is an integral solution. Unfortunately, as shown in Ref. [4], there are problem instances for which this is not true.

Since the RSA for planar graphs was shown to be NP-hard [16], it was conjectured that the RSA for $2d$-graphs was also NP-hard. Shi and Su [5] settled this issue by showing that this problem is an NP-hard problem (and the deci-

DOI: 10.2316/P.2011.757-116

sion version is NP-Complete). The problem is intractable even if all the destinations are in the first quadrant and the source is the origin (point $(0,0)$). Rao, *et. al* presents a simple approximation algorithm for the RSA problem for $2d$-grid networks that generates solutions that are within 100% of optimal (approximation ratio is 2). The algorithm is based on a simple greedy strategy, but the proof that the approximation ratio is 2 is not that simple. In this paper, we give a problem instance for which the approximation ratio is asymptotic to 2. This establishes that the approximation ratio for the algorithm is best possible. The algorithm is based on a greedy bottom-up approach which constructs the tree from the destination nodes to the source node (root), but it is a greedy method in the sense that it considers the destination nodes furthest from the source node first.

Ramnath [6] presents a *top-down* $O(n \log n)$ time 2-approximation algorithm for the RSA problem for $2d$-grid networks. The algorithm is not restricted, as the previous ones, to all nodes being on the first quadrant and the source located in the origin $(0,0)$. The algorithm has also been adapted to the case when there are isothetric obstacles. The algorithm is based on a top-down approach as it constructs the arborescence from the source node s towards the leaves. The algorithm is greedy in the sense that it will cover first the destination nodes closest to the root. We show that this algorithm behaves exactly as the the one in Ref. [4] in the sense that the approximation ratio of 2 is asymptotically best possible. To establish this result we present a problem instance for which the approximation ratio is asymptotic to two.

It is interesting to note that our problem instance that makes the bottom-up algorithm behave poorly can be solved to optimality via the top-down algorithm. Similarly, the instance that makes the top-down algorithm behave poorly can be solved to optimality via the bottom-up approach. This suggests that a better algorithm is just one that generates as its solution the best of the top-down and bottom-up solutions. Both of the above instances can be placed inside a square of size r by r. Now place one of those instances with the origin at point $(0,r)$ and the other with origin at point $(r,0)$. One can show that for this problem instance the combined algorithm has an approximation ratio asymptotic to 1.5. There are other more complex instances such that this combined algorithm behave as bad as either of the algorithms. I.e., there are problem instances for which the solutions asymptotic to 2 times optimal. On average the approximation ratio of the combined algorithm is better than that of the individual algorithms. We experimented with algorithms that use a different strategy at different levels in the process of constructing the sp-multicasting tree, but those algorithms also have the same performance issues.

Lu and Ruan [7] developed a polynomial time approximation scheme for the 2d-grid RSA problem. I.e., they developed a randomized algorithm, that when derandomized, generates a solution such that for every fixed constant $c > 1$, the solution generated uses a number of links that is at most $(1 + 1/c)$ times the number of links in a optimal solution. The time complexity is $O(n^{O(c)} \log n)$. At first glance this seems to be the best possible approximation algorithm for the problem. Unfortunately, this is only true theoretically, as the constants associated with the time complexity bound are enormous and thus the approximation scheme is of no practical significance.

We present in this paper a polynomial time rounding scheme that given a half-integral optimal solution it generates a solution within 1.5 times the value of an optimal solution. Given any instance of the 2d-grid RSA problem we first formulate it as an integer linear programming problem. Then we solve the relaxed linear programming problem and apply our rounding scheme. Our algorithm can be applied to any half-integral optimal solution obtained by any procedure. Our rounding procedure is simple and can be implemented efficiently. We will briefly discuss the integer linear programming formulation and the corresponding relaxed LP problem.

Approximation algorithms for the sp-multicast problem for general and planar graphs have been studied in the past. However, none of the algorithms that take polynomial time generate solutions that are within a fixed percentage of the optimal solution value. This is in sharp contrast to the approximation algorithms for the Steiner tree version of the problem for arbitrarily weighted graphs. In the Steiner tree version one constructs a multicast tree with the fewest links, but the paths from the source to each destination are not necessarily shortest paths in the input graph. For this problem there are well-known polynomial time 2-approximation algorithms.

A problem, closely related to our problem, is the sp-multicast problem defined over the n-*cube* (also called the *hypercube*). There have been many heuristics developed for this version of the problem, but so far the problem does not seem to have a polynomial time constant ratio approximation algorithm. It is important to note that NP-Completeness results for one of these problems does not translate to an NP-Completeness result of the other. Similar observations can be made for approximation algorithms. However, the techniques behind an approximation algorithm for the sp-multicast problem for $2d$-grids may be applicable to the n-cube problem. Since the problem defined over $2d$-grids appears to be simpler, it is important to study it, as the algorithms developed may shed some light for solving more complex versions of the problem.

In Refs. [8, 9, 10] it was established that the decision version of the sp-multicast problem for the n-cube is NP-Complete. These results were extended to establish that the problem over the Chord and the binomial graph network is also NP-Complete [10]. These generalizations have applications to peer-to-peer networks. Numerous heuristics that most of the time generate near-optimum solutions have been developed. However, almost all of these heuristics have been shown to generate arbitrary bad solutions for large sets of problem instances. Because of the similar-

ities to our problem, in what follows we will briefly discuss them.

The simplest top-down algorithm, referred to as the `Oblivious` algorithm by Fujita [11], constructs the tree from the source node s by selecting a minimal subset of intermediate (next) nodes in such a way that all the destinations nodes have a shortest path from s that visits at least one of these intermediate nodes. The process continues from the selected nodes until the sp-multicast tree constructed reaches all the destination nodes.

Lan, Esfahanian, and Ni [12] developed an improved version of the `Oblivious` algorithm that has been referred to as algorithm `LEN` or `Greedy`. The difference between the two algorithms is in the way one selects the intermediary nodes. One selects first an intermediate node from which the maximum number of destination nodes can be reached via a shortest path. Algorithm `Greedy` generates better solutions than the `Oblivious` Algorithm, but there are problem instances for which both of these algorithms generate solutions that are arbitrarily far from optimal [11, 10].

Sheu and Yang [13] modified algorithm `Greedy`. Their algorithm, which we refer to as algorithm `NGrouping`, generates better solutions than algorithm `Greedy`. Their idea is to group together destinations that are neighbors of each other. So instead of having a set of destinations, we have a set of trees of destinations where neighbors nodes are at a distance one from each other. Then one applies algorithm `Greedy` to the root nodes of all the trees.

The `Cluster` algorithm introduced by Lu, Fan, Dou, and Yang [15, 14] is an extension of algorithm `NGrouping`. Again one constructs trees as in the algorithm `NGrouping`, but the trees are more general. A careful analysis of one of Fujita's [11] instances, that makes algorithm `Greedy` perform poorly, can be used to show that algorithms `NGrouping` and `Cluster` generate trees with a number of edges that is arbitrarily large.

Cipriano and Gonzalez [10] present algorithms (`MOverlap`, `BEstimate` and `BUp`) for the n-cube sp-multicasting problem. These algorithms try to avoid the pitfalls of the previously known algorithms. Algorithms `MOverlap` and `BEstimate`, are top-down and estimate the cost of tree before making decisions. Algorithm `BUp` uses the bottom-up approach. All these algorithms take polynomial time, but it is not know whether or not they are constant-ratio approximation algorithms.

In the next section we review two 2-approximation algorithms for the sp-multicast problem. These algorithms correspond to the bottom-up and the top-down algorithms previously developed for the hypercube. Since these algorithms have been shown to generate arbitrarily bad solutions for the hypercube instances, we need to find an alternate approach. One such approach is discussed in Section 3.

2 Tight Approximation Ratios and Experimentation

We begin this section by reviewing the greedy and the bottom-up approximation algorithms presented in Refs. [4, 6] for the $2d$-grid problem. As stated before, we are concentrating on the sp-multicast problem in the $2d$-grid where all the destination nodes are on the first quadrant and the source s is the node with coordinates $(0, 0)$.

Define the diagonal k, for $k \geq 1$, to be the set of all the grid nodes that are at a distance exactly k from the source s. The greedy algorithm, given in Ref. [6] considers all the nodes in diagonal 1, then diagonal 2, and so forth till the last diagonal. When considering diagonal i, a tree has been constructed from s to a subset of the grid nodes in diagonal $i-1$ in such a way that for every destination $d \in D$ there is a shortest path from s to d that includes a path in the tree so far constructed from s to diagonal $i-1$. The tree is now extended to diagonal i by adding a set of edges in such a way that the previous invariant holds for the tree from s to diagonal i. The nodes included from diagonal i are a minimal number of additional nodes and edges.

Figure 1(A) gives an instance for the $2d$-grid and an optimal solution for it. The source node is depicted by a (non-filled) circle and the destination nodes are the smaller (filled) black circles. Figure 1(B) gives the solution generated by the greedy algorithm. One can show that as the set of points increases following the pattern as in Figure 1(A), the approximation ratio is asymptotic to two. This shows that one cannot prove any constant approximation ratio less than two for the greedy method. It is clear for this example, that in order to beat the approximation ratio of 2 one needs to make global decisions, rather than just local ones.

The bottom-up algorithm considers all the nodes in the last diagonal first, then last - 1st diagonal, and so forth till diagonal 1. When considering diagonal i, a forest of trees has been constructed from diagonal $i+1$ to all the destination nodes above diagonal i such that for every destination $d \in D$ above diagonal i there is a shortest path that includes a path in the tree so far constructed from diagonal $i+1$ to the destination nodes above diagonal i. The forest is now extended to diagonal i in such a way that the previous invariant holds for the forest above diagonal $i-1$. The nodes included from diagonal i are a minimal number of additional nodes and edges.

Figure 2(A) depicts a $2d$-grid problem instance and an optimal solution for it. the notation is identical to the one for the previous figure. Figure 2(B) gives the solution generated by the bottom-up algorithm. As the set of points increases following the same pattern as in Figure 2(A), the approximation ratio for the instance grows asymptotic to 2. This shows that one cannot establish any constant approximation ratio less than two for the bottom-up method. It is clear for this example, that in order to beat the approximation ratio of two one needs to make global decisions, rather than just local ones.

We experimented with the above two algorithms, as

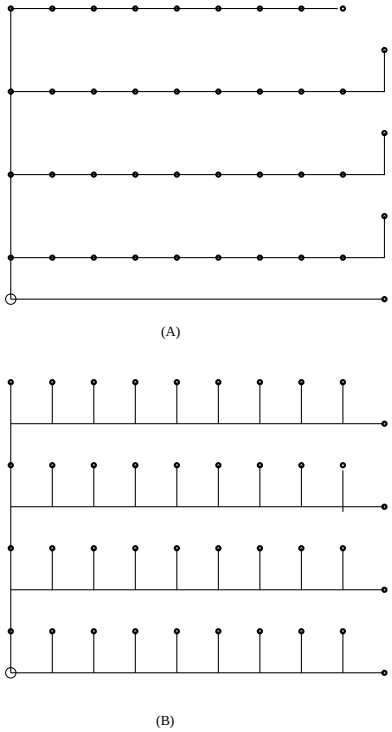

Figure 1. (A) Problem instance and optimal solution. (B) Solution generated by the (greedy) top-down algorithm.

well as with other algorithms, on random problem instances. Table 1 shows the average and worst case approximations for the Bottom-Up, Top-Down, and Hybrid methods. The Hybrid method uses the Bottom-Up approach until the median diagonal and then uses the top down approach. By "Best" we mean the best of the solutions generated by the three algorithms. We use as a lower bound the one presented in [4] that computes the fewest number of points needed in each diagonal by any multicasting tree. The results of the experimentation are clear. As the number of points increases, the average approximation ratio decreases, but the worst approximation ratio increases. This is consistent with the worst case examples discussed in this section.

In the next section we discuss a rounding procedure for a relaxation method that incorporates global information.

3 Rounding Approach

We are concentrating on the sp-multicast problem over the $2d$-grid where all the destination nodes are on the first quadrant and the source s is the node with coordinates $(0, 0)$. In this section we present a global rounding technique that given any half-integral optimal solutions, the algorithm generates a solution with an approximation ratio of at most 1.5. A half-integral optimal solution can be obtain by formulating the problem as an ILP problem and then

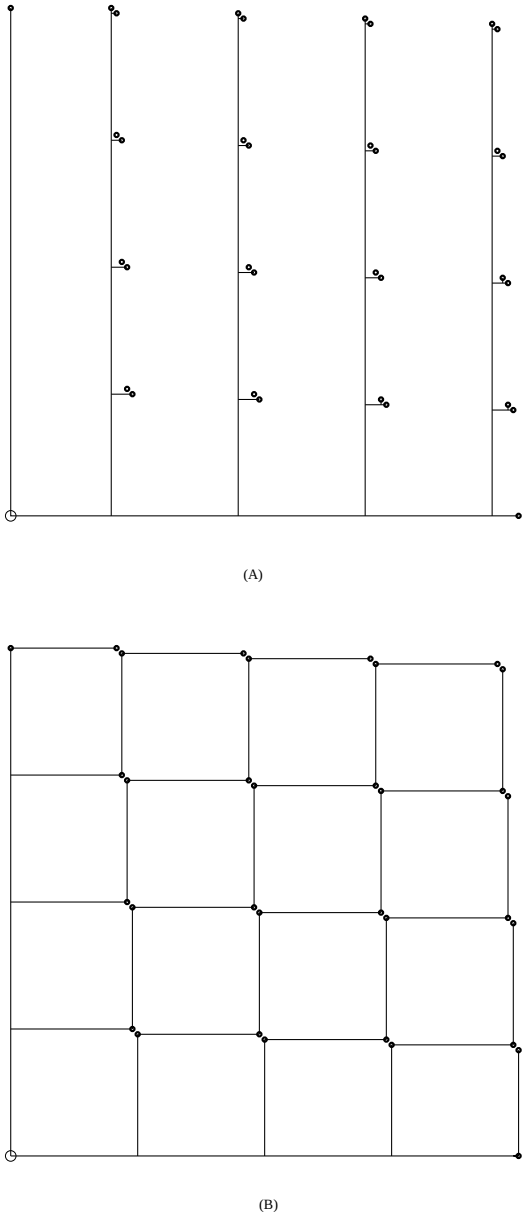

Figure 2. (A) Problem instance and optima solution. (B) Solution generated by the Bottom-Up Algorithm.

Table 1. Experimentation (50000 experiments for each n).

	$n = 10$		$n = 30$	
Method	Worst	Average	Worst	Average
Bottom-Up	14.35	0.85	15.20	2.76
Top-Down	23.10	1.54	21.54	3.91
Hybrid	16.97	0.96	16.81	3.18
Best	9.39	0.58	11.15	2.21
	$n = 50$		$n = 100$	
Method	Worst	Average	Worst	Average
Bottom-Up	12.41	3.43	13.19	4.19
Top-Down	17.45	4.73	13.85	5.57
Hybrid	17.26	3.96	13.51	4.82
Best	10.56	2.98	13.96	3.90

Table 2. Problem instance (15 destinations).

Index	1	2	3	4	5
x	0	25	40	87	93
y	341	197	178	335	261
Index	6	7	8	9	10
x	95	119	143	179	178
y	155	167	235	33	192
Index	11	12	13	14	15
x	180	264	294	294	328
y	143	112	51	84	0

relaxing the integer constraints to obtain a linear programming problem. The LP problem is solved via any of the standards methods. Our LP relaxation is different from the one in [3]. In what follows we will briefly discuss it.

The idea is to define for each destination node D and each grid edge (in the $2d$-grid) a variable to represent whether or not the path connecting s to d goes through that edge. In the ILP formulation the variable will be assigned the value of one or zero. It is one then the path goes through the edge. We also introduce a set of constraints to establish shortest path connectivity to the source node for every destination node. The objective is to add up the maximum value of any of the variables along each edge. In the LP formulation, the values of the variables can be any real number between 0 and 1.

Before we define our ILP formally, we introduce additional notation. The $2d$-grid consists of all the grid points (i, j) for $0 \leq i \leq m$ and $0 \leq j \leq m$, for some positive integer m. The grid edge immediately to left of grid point (i, j) is called *horizontal grid edge* (i, j) and the one immediately below grid point (i, j) is called *vertical grid edge* (i, j). We define variable $X_{i,j}$ for each horizontal grid edge (i, j), and variable $Y_{i,j}$ for each horizontal grid edge (i, j). We define the *region* for every destination node $d \in D$ as the set of all the grid nodes and edges of the rectangle formed by using s and d as its opposite corners. For each destination node $d \in D$ we define the variable $y_{d,\{i,j\}}$, for every vertical grid edge $\{i, j\}$ in the region for node d and the variable $x_{d,\{i,j\}}$, for every horizontal grid edge $\{i, j\}$ in the region for node d. All of our variables are restricted to have the value of zero or one.

For each node d located at grid point (k, l), we add the following constraint.

$$x_{d,\{k,l\}} + y_{d,\{k,l\}} = 1.$$

For each destination node r located at grid point (i, j) in the region for node d, and $d \neq r$, we add the following contraints

$$x_{d,\{i,j\}} = y_{d,\{i,j\}} = 0.$$

For each grid point (i, j) in the region for node d that is not a source or destination node, we add the following contraint

$$x_{d,\{i,j\}} + y_{d,\{i,j\}} = x_{d,\{i,j+1\}} + y_{d,\{i+1,j\}}.$$

We define $D_{i,j}$ as the set of all the destination nodes whose region includes grid point (i, j). For each grid point (i, j) other than the source node we add the following constraints:

$$X_{i,j} \geq Max\{x_{d,\{i,j\}} | d \in D_{i,j}\}$$
$$Y_{i,j} \geq Max\{y_{d,\{i,j\}} | d \in D_{i,j}\}.$$

The objective function value is to minimize

$$\sum_{(i,j)} X_{i,j} + Y_{i,j}. \quad (1)$$

In the LP relaxation all the variables are restricted to have a real value between zero and one.

In order to illustrate our algorithm we will be using an example. The example is given in Table 2 and includes 15 destination nodes. Figure 3 shows a half-integral optimal solution to our example obtaining through our LP formulation. Note that the scale of the figure has been changed in order to see clearly all the steps of our rounding procedure. The small circles are the destination nodes D, and the source is the bottommost point (the large circle). Each thick edge represents a sequence of grid edges for which the optimal half-integral optimal solution to the LP problem assigns the value of 1 to every variable associated with the grid edges ($X_{i,j}$ or $Y_{i,j}$), and the thin ones represent the ones with value $\frac{1}{2}$ (variables $X_{i,j}$ or $Y_{i,j}$). Now lets view Figure 3 as an undirected graph G where the set of vertices represent the source node, the destination nodes plus the Steiner points (points that are not destination nodes where

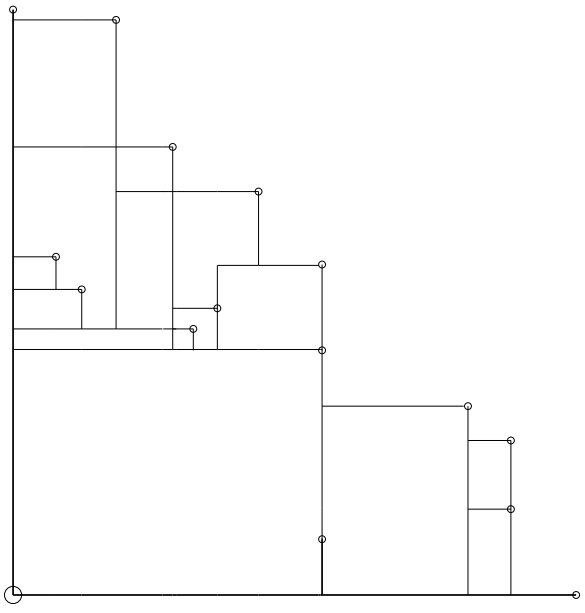

Figure 3. Graph G.

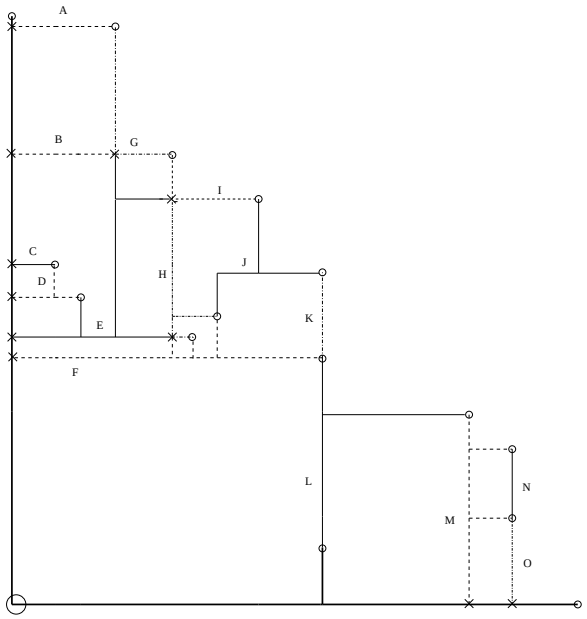

Figure 4. Transition Nodes and Chunks.

three or more line segments intersect). For graph G these vertices are called *source*, *destination*, and *Steiner* vertices. Every edge e has a weight w_e that is a one if it is a thick edge in the graph and $\frac{1}{2}$, otherwise. Every edge e has a length, denoted by l_e, which corresponds to the length of the edge it represents (the length of the edge in Figure 3). A Steiner vertex x is called a *transition* vertex if vertex x has degree four in G or if there is a path from source vertex s to a destination vertex y in G that transitions at vertex x from an edge with a weight of 1 to one with with weight $\frac{1}{2}$, or vice-versa. Figure 4 shows all the transition vertices in G marked with a $\times$ symbol. Later on we explain why some edges are represented by dashed or dash-dotted lines.

Since we begin with a optimal half-integral solution know that $f^*(I) \geq \sum_{e \in G} l_e * w_e$, where $f^*(I)$ is the optimal solution value for the instance I of the sp-multicast problem.

Now the problem is to select all the edges in G with a weight of 1 and a subset of the edges with a weight of $\frac{1}{2}$ in such a way that for every destination vertex d there is at least one path with total length b_d from s to d. Remember that b_d is the length of a shortest path in the $2d$-grid from source node s to destination node d. Let G_1 be the G after deleting all the edges that were not selected. Now let G_2 be G_1 after deleting a subset of (superfluous) edges such that in G_2 there is exactly one path from s to each destination node $d \in D$ with total length b_d. Graph G_2 is a tree and will have all edges with a weight of 1. It is the solution we generate to the instance of the $2d$-grid sp-multicast problem, i.e., it is the multicast tree we generate. This solution has total edge length (or objective function value) equal to $\hat{f}(I) = \sum_{e \in G_2} l_e$.

Before we present our rounding algorithm to generate the graph G_2 such that $\hat{f}(I) \leq \frac{1}{2} f^*(I)$, we need to introduce additional notation. We partition the set of edges with weight $\frac{1}{2}$ in G into chunks. A *chunk* is a maximal set of nodes and edges with weight $\frac{1}{2}$ in G such that all of its destination vertices and transition vertices are its leaf vertices, and all its leaves are destination or transition vertices. Figure 4 shows the different chunks. Each chunk is identified by a different letter and represents a set of adjacent edges drawn using the same type of lines (continuous, dashed, or dash-dotted). The bottommost left vertex of each chunk is called the *root* of the chunk, and the remaining leaves are called the *end* vertices of the chunk.

It is easy to see that each end vertex is the end vertex of exactly two chunks. The edges of two different chunks overlap only at root or end vertices. Therefore it is possible two draw a closed and continuous curve around the root and end vertices of each chunk in such a way that no two curves of two different chunks overlap at a point different from a root or end vertex. Now construct the graph H as follows. Each chunk is represented by a vertices in H (in our example the vertex corresponding to chunk c is assigned the letter associated with the chunk c) and there is an edge between two vertices in H, if the corresponding chunks include the same end node. Since the chunks only overlap at destination nodes and transition nodes, it then follows that graph H is a planar graph. Figure 5 shows the graph H constructed for the chunks given in Figure 4. The *cost*, denoted by c_v, of vertex v in H is the sum of the length of the edges in the chunk corresponding to vertex v in graph H.

A *coloring* for a planar graph is an assignment of a color to each vertex in such a way that no two adjacent vertices are assigned the same color. It is well known (see for example Ref. [16]) that every planar can be colored with at

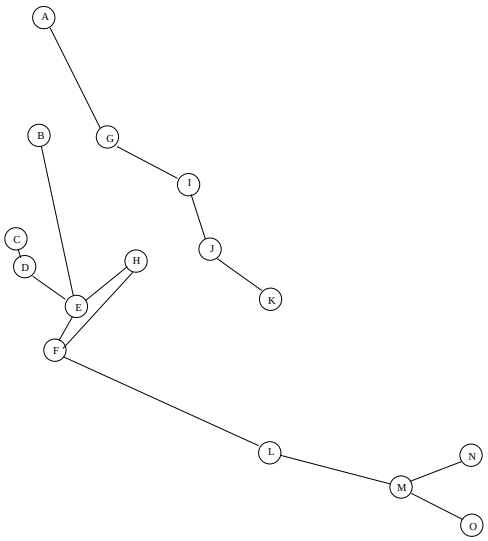

Figure 5. Graph H for the graph in Figure 4.

most four colors. Furthermore, there are well known linear time algorithms to construct a four coloring of any given planar graph. Applying the algorithm to graph H results in the set of vertices partitioned into four sets S_1, S_2, S_3, and S_4, where S_i consists of all the vertices assigned color i. Now define s_i as the sum of the cost of the nodes in set S_i, i.e., $s_i = \sum_{j \in S_i} c_j$. Assume without loss of generality that $s_1 \leq s_2 \leq s_3 \leq s_4$. Now the edges selected in G to form G_2 are all the edges in the chunks represented by the vertices in the sets S_1, S_2, and S_3 plus all the edges with weight 1 in G. Let T be the set of edges with weight 1 in G and t denotes the sum of the length of the edges in T.

Since G is an optimal relaxed solution for the integer linear programming solving the sp-multicast problem, it follows that

$$f^*(I) \geq t + \frac{1}{2}(s_1 + s_2 + s_3 + s_4). \tag{2}$$

Now, the solution we generate is such that

$$\hat{f}(I) \leq t + s_1 + s_2 + s_3.$$

This is equivalent to

$$\hat{f}(I) \leq t + \frac{3}{4}(s_1 + s_2 + s_3) + \frac{1}{4}(s_1 + s_2 + s_3) \tag{3}$$

Since $s_1 \leq s_2 \leq s_3 \leq s_4$, we know that

$$s_4 \geq \frac{1}{3}(s_1 + s_2 + s_3).$$

Substituting in Equation 2 we know that

$$\hat{f}(I) \leq t + \frac{3}{4}(s + 1 + s_2 + s_3 + s_4). \tag{4}$$

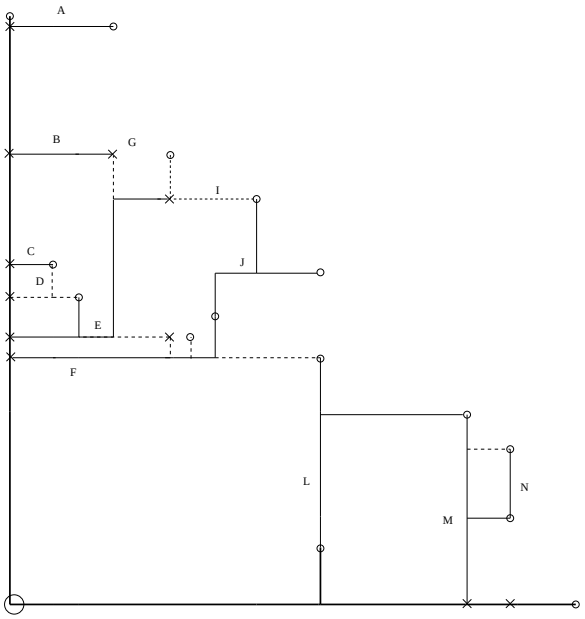

Figure 6. Solid edges represent a final sp-multicat tree for our example.

Combining Equations 4 and 2, we know that we know that

$$\hat{f}(I) \leq 1.5 f^*(I). \tag{5}$$

We summarize our result in the following theorem.

Theorem 1: Given any half-integral optimal solution to the sp-multicast tree problem one can round it and generate an sp-multicast tree with objective function value that is at most 1.5 times the optimal solution value. The algorithm takes polynomial time with respect to the number of destination nodes.

The proof for the approximation ratio follows the above arguments. For brevity the proof for the time complexity bound is omitted.

Now, for some problem instances one may be able to color graph H with two or three colors. In the former case one can construct an optimal solution to the sp-multicast problem. For the case of three colors, we use a similar approach and obtain a better approximation ratio, $\hat{f}(I) \leq \frac{4}{3} f^*(I)$.

In our example it is possible to color with three colors the graph H. One such coloring is given in Figure 4 by interpreting the solid, dashed, dash-dotted lines for the chunks as their colors. Figure 6 shows the case when the dash-dotted lines are eliminated from Figure 4. The dashed lines in Figure 6 do not correspond to to the dashed lines in Figure 4. The dashed lines in Figure 6 form a subset of superfluous edges which can be eliminated. Therefore the solid lines (thick and thin) form the sp-multicast tree generated by our rounding algorithm. Figure 7 shows the case when the solid (thin) lines are eliminated from Figure 4. The dashed lines in Figure 7 do not correspond to to

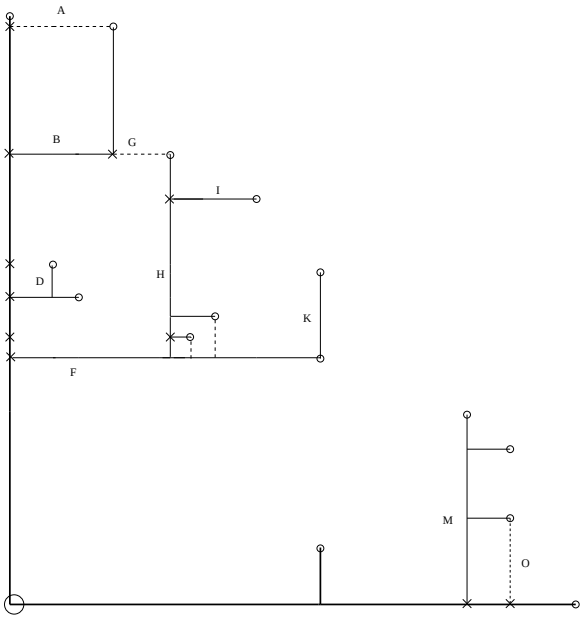

Figure 7. Solid edges represent a final sp-multicat tree for our example.

the dashed lines in Figure 4. The dashed lines in Figure 7 form a subset of superfluous edges which can be eliminated. Therefore the solid lines (thick and thin) form the sp-multicast tree generated by our rounding algorithm.

4 Conclusion

The most interesting open problem is to show that every $2d$-grid sp-multicast tree problem allows a half-integral optimal solution. Our experimental evaluation confirms this fact, but we have not been able to prove it. Another interesting open problem is to slightly perturb the destination nodes to obtain an integer solution which can then be transformed to a solution to the original problem. We have shown, through extensive experimentation, that the solutions produced by our rounding technique are for the most part optimal or near optimal as the graph H is normally either empty or it is a bipartite graph for which our rounding generates an optimal solution.

References

[1] Ladeira de Matos, R. R., A Rectilinear Arborescence Problem, Ph.D. Dissertation, University of Alabama, 1979.

[2] Nastansky, L., Selkow, S. M., and Stewart, N. F., "Cost-minimal trees in directed acyclic graphs, *Z/ Oper. Res.* 18, 59 – 67, 1974.

[3] Trubin, V. A., "Subclass of the Steiner problems on a plane with rectilinear metric, Cybernetics, 21 320 - 322, 1985, translated from Kibernetika 21, 33 - 40, 1985.

[4] Rao, S. K., Sadayappan, P., Hwang, F. K., and Shor, P.W., "The rectilinear Steiner arborescence problem," *Algorithmica*, 7, 277 – 288, 1992.

[5] Shi, W., and Su, C., "The rectilinear Steiner Arborescence problem is NP-Complete," *SIAM J. Comput.*, 35(3), 729 – 740, 2006.

[6] Ramnath, S., New Approximations for the rectilinear Steiner arborescence problem, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, (22) 7, 859 – 869, 2003.

[7] Lu, B., and Ruan, L., "Polynomial Time Approximation Scheme for the Rectilinear Steiner Tree Arborescence Problem," *Journal of Combinatorial Optimization*, 4, 357 – 363, 2000.

[8] Choi, H.-A., and Esfahanian, A. H., "On Complexity of a Message-Routing Strategy for Multicomputer Systems," *Lecture Notes in Computer Science*, R.H. Mohring (Ed.), 484, 170 – 181, Springer-Verlag, 1991.

[9] Choi, H.-A., Esfahanian, A. H., and Houck, B., "Optimal Communication Trees with Application to Hypercube Multiprocessors," *Graph Theory, Combinatorics, and Applications*, Alavi, Y, Chartrand, G., Oellermann, O. R., and Schwenk, A. J. (Eds.), 1, 245 – 264, 1991.

[10] Cipriano, C.C., and Gonzalez, T. F., Multicasting in the Hypercube, Chord and Binomial Graphs, Technical Report 2009-09, UCSB, May 2009.

[11] Fujita, S., A Note on the Size of a Multicast Tree in Hypercubes, *Information Processing Letters*, Vol 54, pp. 223 – 227, 1995.

[12] Lan, Y., Esfahanian, A. H., and Ni, L. M., Multicast in Hypercube Multiprocessors, *Journal of Parallel and Distributed Computing*, Vol 8, pp. 30 – 41, 1990.

[13] Sheu, S.-H. and Yang, C.-B., Multicast Algorithms for Hypercube Multiprocessors, *Journal of Parallel and Distributed Computing*, Vol 61, pp. 137 – 149, 2001.

[14] Lu, SL and Yang, XD, A Clustering Model for Multicast on Hypercube Network, GPC-2008, *LNCS*, Vol 5036, pp. 211 – 221, 2008.

[15] Lu, S., Fan, BH, Dou, Y., and Yang, XD, Clustering Multicast on Hypercube Network, HPCC-2006, *LNCS*, Vol 4208, pp. 61 – 70, 2006.

[16] Garey, M.R., and Johnson D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.