LA-UR 94-4088

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE: **BICRITERIA NETWORK DESIGN PROBLEMS**

AUTHOR(S): M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz
H. B. Hunt III

SUBMITTED TO: 22nd International Colloquium on Automata, Languages and Programming
June, 1995

DISCLAIMER

Los Alamos

Los Alamos National Laboratory
Los Alamos New Mexico 87545

Bicriteria Network Design Problems

(Preliminary Version)

M. V. Marathe* R. Ravi† R. Sundaram‡ S. S. Ravi[1] D. J. Rosenkrantz[1,3]
H.B. Hunt III[1,2]

Abstract

We study several *bicriteria* network design problems phrased as follows: given an undirected graph and two minimization objectives with a budget specified on one objective, find a subgraph satisfying certain connectivity requirements that minimizes the second objective subject to the budget on the first. Define an (α, β)-approximation algorithm as a polynomial-time algorithm that produces a solution in which the first objective value is at most α times the budget, and the second objective value is at most β times the minimum cost of a network obeying the budget on the first objective. We present the first approximation algorithms for bicriteria problems obtained by combining classical minimization objectives such as the total edge cost of the network, the diameter of the network and a weighted generalization of the maximum degree of any node in the network.

We first develop some formalism related to bicriteria problems that leads to a clean way to state bicriteria approximation results. Secondly, when the two objectives are similar (e.g., both the objectives are to minimize the total edge cost of the network) but only differ based on the cost function under which they are computed (e.g., two different costs are specified on edges and the two objectives are to minimize the total edge cost based on the two different costs), we present a general parametric search technique that yields approximation algorithms by reducing the problem to one of minimizing a single objective of the same type. Thirdly, we present an $O(\log n, \log n)$-approximation algorithm for finding a diameter-constrained minimum cost spanning tree of an undirected graph on n nodes generalizing the notion of shallow, light trees and light approximate shortest-path trees that have been studied before. Finally, for the class of treewidth-bounded graphs, we provide pseudopolynomial-time algorithms for a number of bicriteria problems using dynamic programming. These pseudopolynomial-time algorithms can be converted to fully polynomial-time approximation schemes using a scaling technique.

Keywords: Approximation algorithms, Bicriteria problems, Spanning trees, Network design, Combinatorial algorithms.

*Los Alamos National Laboratory P.O. Box 1663, MS M986, Los Alamos NM 87545. Email: madhav@gardener.lanl.gov. Research supported by the Department of Energy under Contract W-7405-ENG-36.

†Dept. of Computer Science, Princeton University, Princeton, NJ 08544-2087. Email: ravi@cs.princeton.edu. Research supported by a DIMACS postdoctoral fellowship.

‡Dept. of Computer Science, MIT LCS, Cambridge MA 02139. Email: koods@theory.lcs.mit.edu. Research supported by DARPA contract N0014-92-J-1799 and NSF 92-12184 CCR.

[1]Dept. of Computer Science, University at Albany - SUNY, Albany, NY 12222. Email:{madhav, ravi, djr, hunt}@cs.albany.edu

[2]Supported by NSF Grants CCR 89-03319.

[3]Supported by NSF Grants CCR 90-06396.

1 Introduction

Several fundamental problems in the design of communication networks can be modeled as finding a network obeying certain connectivity constraints. The goal in such network design problems is usually to minimize a certain measure of cost associated with the network. The cost may reflect the price of synthesizing the network, or it may reflect the maximum delay between the ends of the network or even some measure of the vulnerability of the network. Examples of such cost measures are the total edge cost, the diameter and maximum degree of the network respectively. Finding a network of sufficient generality minimizing even one of these measures is often NP-hard [17]. Moreover, in applications that arise in real-life situations, it is often the case that the network to be built is required to minimize more than one cost measure simultaneously. In this paper, we consider bicriteria problems motivated by practical instances arising in the design, analysis and synthesis of communication networks.

We first develop a formalism for bicriteria problems and their approximations. A typical bicriteria problem, $(\mathcal{A}, \mathcal{B})$, is defined by identifying two minimization objectives of interest from a set of possible objectives. The problem specifies a budget value on the first objective, $\mathcal{A}$, and seeks to find a network having minimum possible value for the second objective, $\mathcal{B}$, such that this network obeys the budget on the first objective. An (α, β)-approximation algorithm is defined as a polynomial-time algorithm that produces a solution in which the first objective value is at most α times the budget, and the second objective value is at most β times the minimum for any solution obeying the budget on the first objective. As an example, consider the following *diameter-bounded minimum spanning tree problem* or (Diameter, Total cost) bicriteria problem: given an undirected graph $G = (V, E)$ with two different integral nonnegative weights f_e (modeling the cost) and g_e (modeling the delay) for each edge $e \in E$, and an integral bound B (on the total delay), find a minimum f-cost spanning tree such that the diameter of the tree under the g-costs (the maximum delay between any pair of nodes) is at most B.

There are two natural alternative ways of formulating general bicriteria problems, one where we impose the budget on the first objective and seek to minimize the second and two, where we impose the budget on the second objective and seek to minimize the first. We show that an (α, β)-approximation algorithm for one of these formulations naturally leads to a (β, α)-approximation algorithm for the other. Thus we show that there is a clean way to state bicriteria approximation results.

2 Summary of results and related research

We summarize our results in Table 1. The table contains the performance ratios for finding spanning trees under different pairs of minimization objectives. All results in the table extend to finding Steiner trees with at most a constant factor worsening in the performance ratios. We omit elaboration on these extensions. The horizontal entries denote the budgeted objective. For example the entry in row i, column j denotes the performance guarantee for the problem of minimizing objective j with a budget on the objective i. As a result of the equivalence mentioned in the previous section, the table is symmetric, i.e. entry (i, j) is identical to entry (j, i). For each of the problems catalogued in the table, two different costs are specified on the edges of the input undirected graph: the first objective is computed using the first cost function and the second objective, using the second cost function.

In the table, the "Degree" objective denotes the maximum degree of any node in the spanning tree; the entry (Degree, Degree) however refers to a generalization to a weighted variant based on two cost functions defined on the edges. This weighted variant of the degree objective is defined as the maximum over all nodes, of the sum of the costs of the edges incident on the node in the tree.

When all edges in the graph have unit weight, this reduces to the maximum degree of any node in the tree. The "Diameter" objective is the maximum distance between any pair of nodes in the tree. The "Total cost" objective is the sum of the costs of all the edges in the tree.

Cost Measures	Degree	Diameter	Total Cost
Degree	$(O(\log n), O(\log n))^*$	$(O(\log n), O(\log n))$[30]	$(O(\log n), O(\log n))$[28]
Diameter	$(O(\log n), O(\log n))$[30]	$(1+\gamma, 1+\frac{1}{\gamma})^*$	$(O(\log n), O(\log n))^*$
Total Cost	$(O(\log n), O(\log n))$[28]	$(O(\log n), O(\log n))^*$	$(1+\gamma, 1+\frac{1}{\gamma})^*$

Table 1. Performance Guarantees for finding spanning trees in an arbitrary graph on n nodes. Asterisks indicate results obtained in this paper. $\gamma > 0$ is a fixed accuracy parameter.

Similar objectives

The diagonal entries in the table follow as a corollary of the general result that is proved using a parametric search algorithm.

Theorem 2.1 Let $\mathcal{P}$ denote a single criterion minimization problem defined on a graph G with costs h associated with the elements of G and let $\gamma > 0$ be a fixed accuracy parameter. Assume that there exists a ρ-approximation algorithm for $\mathcal{P}$. Then the bicriteria problem $\mathcal{P}_2$ defined on G by specifying two different costs f and g on the elements of G and the two objectives being minimizing the objective of $\mathcal{P}$ under the two different costs f and g has a $((1+\gamma)\rho, (1+\frac{1}{\gamma})\rho)$-approximation algorithm.

The diagonal entries in the table correspond to such bicriteria problems in which the two objectives are similar but only differ in the cost function on the edges under which they are computed; For example, the (Total Cost, Total Cost) problem is the following: given an undirected graph $G = (V, E)$ with two different integral nonnegative weights f_e and g_e for every edge $e \in E$, and an integral bound B, find a minimum f-cost spanning tree such that the total cost of the tree under the g-costs is at most B. Thus, both objectives correspond to the total cost of the tree but under two different cost functions g and f. For such problems, we introduce a parametric search method on a hybrid cost function $h_e = f_e + \mu g_e$ on the edges e, such that the single objective problem solved using the hybrid cost h for an appropriately chosen μ yields a good approximation for both the original objectives. For example, for the minimum-cost spanning tree problem, using Theorem 2.1 with $\rho = 1$ gives the (Total Cost, Total Cost) result in our table. The result for (Diameter, Diameter) follows similarly from known exact algorithms for minimum diameter spanning trees [13, 29].

The entry for (Degree, Degree) corresponds to the following problem: given an undirected graph $G = (V, E)$ with two different integral nonnegative weights $c_1(e)$ and $c_2(e)$ for every edge $e \in E$, and an integral bound D, find a spanning tree of minimum weighted degree under c_2 such that the weighted degree of the tree under the c_1-costs is at most D. Recall that the weighted degree of the tree under a cost c is the maximum over all nodes, of the sum of the c-costs of the edges incident on the node in the tree. An $O(\log n)$-approximation algorithm for the weighted degree problem was presented by Ravi et al. in [28]. Hence the entry for (Degree, Degree) follows from Theorem 2.1 and the results in [28].

Theorem 2.1 is quite general in that it allows translation of a single objective approximation for *any* network design problem to one for two similar objectives of the same type with a factor-of-two worsening in the performance guarantee. Consider, for example, the conjunction of this theorem with the results of Goemans et al. [18]. This leads to a host of bicriteria approximation results when two costs are specified on edges for finding minimum-cost generalized steiner trees, minimum

k-edge connected subgraphs, and other network design problems specified by weakly supermodular functions introduced in that paper. Similarly we may specify two costs on every node in the graph and get logarithmic approximations for the minimum node-cost Steiner trees using the result of Klein and Ravi [22]. As another example, with two costs specified on edges, and an input number k, we can use the result of Awerbuch et al. [6] to obtain $(O(\log^2 k), O(\log^2 k))$ approximations for the minimum cost (under both functions) tree spanning at least k nodes.

Ravi et al. [28] studied the degree-bounded minimum cost spanning tree problem. They provided an approximation algorithm with performance guarantee $(O(\log n), O(\log n))$. The problem of finding a degree-bounded minimum diameter spanning tree was studied by Ravi [30] in the context of finding good broadcast networks. He provided an approximation algorithm for the first problem with performance guarantee $(O(\log n), O(\log n))$ with an extra additive term of $O(\log^2 n)$ for the degree.

The (Diameter, Total cost) entry in Table 1 corresponds to the diameter-constrained minimum spanning tree problem introduced earlier. This problem arises naturally in the design of networks used in multicasting and multimedia applications [15, 16, 20, 23, 24, 25]. It is known [17] that this problem is NP-hard. In the special case when the two cost functions are identical, i.e., $f_e = g_e$ for all edges e, the diameter-bounded minimum spanning tree problem reduces to finding a spanning tree that has simultaneously small diameter (i.e., shallow) and small total cost (i.e., light), both under the same cost function. Awerbuch, Baratz and Peleg [7] showed how to compute in polynomial-time such *shallow, light trees* while Khuller, Raghavachari and Young [21] studied an extension called *Light, approximate Shortest-path Trees* (LASTs). Kadaba and Jaffe [20] and Kompella et al. [23] considered the general diameter-bounded minimum spanning tree problem and presented heuristics without any guarantees. We present the first approximation algorithm for this problem; the performance ratios for both objectives are logarithmic.

Treewidth-bounded graphs

We also study the bicriteria problems mentioned above for the class of treewidth-bounded graphs. Robertson and Seymour in their seminal series of papers [31] introduced and developed the notion of treewidth. Many hard problems have exact solutions when attention is restricted to the class of treewidth-bounded graphs and much work has been done in this area (see [1, 2, 3, 4, 5, 9, 10, 11, 12, 14]). Examples of treewidth-bounded graphs include trees, series-parallel graphs and bounded-bandwidth graphs. Independently, Bern, Lawler and Wong [9] introduced the notion of decomposable graphs. Later, it was shown [5] that the class of decomposable graphs and the class of treewidth-bounded graphs are one and the same.

To the best of our knowledge this is the first time that bicriteria problems have been studied for the class of treewidth-bounded graphs. We use a dynamic programming technique to show that for any class of decomposable graphs (or treewidth-bounded graphs) , there are either polynomial-time (when the problem is in P) or pseudopolynomial-time algorithms (when the problem is NP-complete) for several of these bicriteria problems. We then show how to convert these pseudopolynomial-time algorithms into fully polynomial-time approximation schemes using a general scaling technique. We summarize our results for this class of graphs in Table 2. As before, the horizontal entries denote the budgeted objective.

Our results for treewidth-bounded graphs have an interesting application in the context of finding optimum broadcast schemes. Kortsarz and Peleg [26] gave $O(\log n)$-approximation algorithms for the minimum broadcast time problem for series-parallel graphs. Using our results for the (Degree, Diameter) case along with the techniques in [30], we can obtain an $O(\frac{\log n}{\log \log n})$-approximation algorithm for this problem for the class of treewidth-bounded graphs (series-parallel graphs have a treewidth of 2), improving and substantially generalizing the results of Kortsarz and Peleg. The

Appendix includes a proof sketch of this result.

Cost Measures	Degree	Diameter	Total Cost
Degree	(open) pseudopoly	(open) pseudopoly	poly-time
Diameter	(open) pseudopoly	(weak NP-hard) pseudopoly	(weak NP-hard) pseudopoly
Total Cost	poly-time	(weak NP-hard) pseudopoly	(weak NP-hard) pseudopoly

Table 2. Bicriteria spanning tree results for treewidth-bounded graphs

In the next section, we discuss our approximations for general graphs. We then derive the results for treewidth-bounded graphs in Section 4.

3 Approximation algorithms

In this section we discuss our approximation algorithms for general graphs. Our approximation algorithms are based primarily on three techniques. We briefly comment on these techniques next. The first technique is a variant of binary search that is used to prove equivalence of bicriteria formulations. Our second technique is parametric search that is used to solve the case of similar objectives. Finally, we solve the diameter-constrained minimum spanning tree problem using a logarithmic number of instances of perfect matching problems to iteratively build up a solution.

3.1 Equivalence of bicriteria formulations

We formulate a general bicriteria problem in network design as follows: Given a graph G and two integral cost functions, say c and d, defined on a class $\mathcal{S}$ of subgraphs of G (e.g., spanning trees of G), and a bound on the value of one of the costs (say C for the c-cost), find a subgraph in $\mathcal{S}$ that has cost at most C under the cost function c and the minimum possible cost under d given this restriction on the c-cost.[4] We call this the C-bounded minimum-d-cost subgraph problem. The alternative formulation would be to use a bound D on the d-cost of the solution and ask for a minimum-c-cost subgraph under this restriction. This alternative formulation may be termed the D-bounded minimum-c-cost subgraph problem.

Note that such bicriteria problems are meaningful only when the two criteria are "hostile" with respect to each other in that the objective of minimizing one is incompatible with that of minimizing the other. A good example of such hostile objectives are the degree and the total edge cost of a spanning tree in an unweighted graph [28]. The notion of hostility between criteria can be formalized by defining two minimization criteria to be hostile whenever the minimum value of one of the objectives is monotonically nondecreasing as the budget on the value of the other objective is decreased.

Define a (ρ_c, ρ_d)-approximation algorithm for the C-bounded minimum-d-cost subgraph problem to be a polynomial-time algorithm that delivers a subgraph from $\mathcal{S}$ that has c-cost at most ρ_c times the bound C, and d-cost at most ρ_d times the minimum d-cost of any subgraph in $\mathcal{S}$ with c-cost at most C.

Theorem 3.1 The existence of a (ρ_c, ρ_d)-approximation algorithm for the C-bounded minimum-d-cost subgraph problem implies the existence of a (ρ_d, ρ_c)-approximation algorithm for the D-bounded minimum-c-cost subgraph problem.

[4] We use the term "cost under c" or "c-cost" in this section to mean value of the objective function computed using c, and not to mean the total of all the c costs in the network.

Proof: The proof of the theorem uses binary search on the range of values of the c-cost with an application of the given approximation algorithm at each step of this search. Suppose we are given $\mathcal{A}$, a (ρ_c, ρ_d)-approximation algorithm for the C-bounded minimum-d-cost subgraph problem. Given a D-bounded minimum-c-cost subgraph problem, we wish to derive a (ρ_d, ρ_c)-approximation algorithm for this problem.

To do this, we initially compute rough upper and lower bound estimates C_{hi} and C_{lo} on the c-cost of a D-bounded subgraph. By using binary search in $[C_{lo}, C_{hi}]$, we find a C' in this range such that

(1) the algorithm $\mathcal{A}$ with c-cost bound C' returns a subgraph from $\mathcal{S}$ with d-cost greater than $\rho_d \cdot D$, and

(2) $\mathcal{A}$ with c-cost bound $C' + 1$ returns a subgraph with d-cost at most $\rho_d \cdot D$.

Suppose the minimum-c-cost of a D-bounded subgraph in $\mathcal{S}$ is C_D. As a consequence of (1) above and the performance guarantee of the approximation algorithm $\mathcal{A}$ we get that $C' < C_D$. Hence we have that $C' + 1 \leq C_D$ by our definition of hostility and the fact that the cost function c is integral. By (2) above we get that the approximation algorithm $\mathcal{A}$ with input $C' + 1$ delivers a subgraph in $\mathcal{S}$ with d-cost at most $\rho_d \cdot D$ and the c-cost of this subgraph is at most $\rho_c(C' + 1)$. This latter quantity is in turn is at most $\rho_c \cdot C_D$. Thus we have a (ρ_d, ρ_c)-approximation algorithm for the D-bounded minimum-c-cost subgraph problem as claimed in the Theorem. □

Note however that the (ρ_d, ρ_c)-approximation algorithm constructed in the proof above may not be strongly polynomial since it depends on the range of the c-costs. But it is a polynomial-time algorithm since its running time is linearly dependent on $\log B$ where B is the value of the largest c-cost.

3.2 Parametric search for approximations of similar objective functions

In this section, we present approximation algorithms for a broad class of bicriteria problems where both the objectives in the problem are of the same type (e.g., both are total edge costs of some network computed using two different costs on edges, or both are diameters of some network calculated using two different costs etc.). Before we present our approximation algorithm, we state the following hardness result that can be derived by a reduction from the Partition problem [17]. A proof is sketched in the Appendix.

Theorem 3.2 The (Diameter, Total cost), (Diameter, Diameter) and the (Total cost, Total cost) spanning tree problems are NP-hard even for series-parallel graphs.

We now present the approximation algorithm used to prove Theorem 2.1. We illustrate the proof of the theorem by considering the case (Total cost, Total cost) in Table 1. In this problem, we are given two costs f_e and g_e on the edges $e \in E$ of the input graph $G = (V, E)$. We are also given a budget B on the total cost under g of the spanning tree T. Assume for now that the magnitude of costs f on the edges are polynomial in the size of the input graph.

Let OPT denote the minimum cost of the tree under f which obeys the restriction that its cost under g is at most B. For any tree T and cost function f, we use $COST_f(T)$ to denote the cost of T under f. To simplify the analysis, we assume that γ divides OPT. This can be enforced by scaling both the cost functions f and g by γ.

Algorithm Two-Cost(G, f, g, B, γ)

Input: A graph $G(V, E)$ with two cost functions f and g on the edges, a budget B on the cost of the spanning tree under g and a performance requirement $\gamma > 0$.

Output: A spanning tree T of G such that the cost of T under g is no more than $(1+\frac{1}{\gamma})B$ and the cost of T under f is no more than $(1+\gamma)OPT$.

1 Initialize $C := 1$
2 **While** (Test(C) = NO) **do**
3 $C := C + 1$
4 Output the tree T computed by **Test** as the solution.

Procedure Test(C)

1 $\mu := \frac{C}{B}$.
2 Compute a new cost function h on the edges $e \in E$ as follows: $h(e) := f(e) + \mu g(e)$.
3 Compute a minimum spanning tree T in the graph $G(V, E)$ under the cost function h.
4 **If** $COST_h(T) \leq (1+\gamma)C$ **then** output YES **else** output NO.

Lemma 3.3 The function $\mathcal{R}(C) = \frac{COST_h(MST)}{C}$ as C takes increasing integral values from $1, 2, 3, \ldots$ is monotone nonincreasing.

Proof: Suppose for a contradiction that for two integral values of C, say C_1 and C_2 with $C_1 < C_2$, we have that $\mathcal{R}(C_1) < \mathcal{R}(C_2)$. Let T_1 and T_2 denote minimum spanning trees of G under h when $C = C_1$ and $C = C_2$ respectively. For $i \in \{1, 2\}$, let F_i and G_i denote the costs of the tree T_i under f and g respectively. Thus, we have that $\mathcal{R}(C_i) = \frac{F_i}{C_i} + \frac{G_i}{B}$ for $i \in \{1, 2\}$.

Consider the cost under h of the spanning tree T_1 when $C = C_2$. By the definition of F_1 and G_1, it follows that the cost of T_1 is $F_1 + \frac{G_1 \cdot C_2}{B}$. Thus the value of $\mathcal{R}(C_2)$ is at most this cost divided by C_2 which is $\frac{F_1}{C_2} + \frac{G_1}{B}$. This in turn is less than $\frac{F_1}{C_1} + \frac{G_1}{B}$, since $C_1 < C_2$. But $\frac{F_1}{C_1} + \frac{G_1}{B}$ is exactly $\mathcal{R}(C_1)$ contradicting the assumption that $\mathcal{R}(C_1) < \mathcal{R}(C_2)$. □

Theorem 3.4 Let C' be the value of C when the Test procedure outputs YES. Let $T_{C'}$ denote the corresponding solution tree. Then $COST_f(T_{C'}) \leq (1+\frac{1}{\gamma})OPT$ and $COST_g(T_{C'}) \leq (1+\gamma)B$.

Proof: First consider the value of $COST_h(T)$ when $C = C^* = \frac{OPT}{\gamma}$ (Note that C^* is integral by assumption). This is at most $OPT + \frac{C^* B}{B} \leq OPT + C^* = (1+\gamma)C^*$. Thus the value of $\mathcal{R}(C^*) = \frac{COST_h(MST)}{C^*} \leq 1 + \gamma$.

Since $\mathcal{R}(C^*) \leq 1 + \gamma$, the function $\mathcal{R}(C)$ is monotone nonincreasing by Lemma 3.3, and C' is the least integer such that $\mathcal{R}(C') \leq 1 + \gamma$, we have that $C' \leq C^*$. It is now easy to verify that the following inequalities hold for $COST_f(T_{C'})$ and $COST_g(T_{C'})$.

$$COST_f(T_{C'}) \leq COST_h(T_{C'}) \leq OPT + \frac{C'}{B}B \leq OPT + C^* \leq (1+\frac{1}{\gamma})OPT$$

$$\frac{C'}{B}COST_g(T_{C'}) \leq COST_h(T_{C'}) \leq C'(1+\gamma)$$

The second chain of inequalities implies that $COST_g(T_{C'}) \leq (1+\gamma)B$. □

We can obtain a better running time by doing a binary search for C in **Algorithm Two-cost** thus using only $O(\log F)$ calls to the polynomial-time test procedure **Test**, where F denotes the ratio of maximum edge cost to the minimum edge cost under f. This allows us to weaken the assumption that the costs under f are polynomial to the condition that the ratio of the maximum to minimum edge cost under f be polynomially bounded. In the most general case, we can obtain polynomial running time for the whole algorithm by scaling the f-costs such that the ratio of the maximum to

the minimum edge cost under f is at most a polynomial, $P(n)$, in the size, n, of the input graph. The scaling results in an inaccuracy of the estimate of the total f-cost by a factor proportional to $\frac{n}{P(n)}$ which would result in a multiplicative factor of $1 + \frac{n}{P(n)}$ in the performance ratio for the f-cost of the tree. This factor tends to one asymptotically and hence is omitted in the statement of the theorem. This completes the proof of this example of Theorem 2.1.

3.3 The algorithm for diameter-constrained trees

In this section, we study the diameter-bounded minimum spanning tree problem. To motivate what kind of approximations we can hope to look for, we present a hardness result on approximating the diameter-bounded Steiner tree problem.

Theorem 3.5 Unless $NP \subseteq DTIME(n^{\log \log n})$, given an instance of the diameter-bounded minimum cost Steiner tree problem with k sites, there is no polynomial-time approximation algorithm that outputs a Steiner tree of diameter at most the bound D, and cost at most R times that of the minimum cost diameter-D Steiner tree, for $R < \log k/8$.

The above theorem is proved using a reduction from set cover and invoking known hardness results [8, 27] on approximating set cover. A sketch is provided in the Appendix. On the positive side, we prove the following theorem.

Theorem 3.6 There is a polynomial-time algorithm that, given an undirected graph G on n nodes with nonnegative integral costs d and c on its edges, a bound D, and a fixed $\epsilon > 0$, constructs a spanning tree of G of diameter at most $2\lceil \log_2 n \rceil D$ under the d-costs and of total c-cost at most $(1+\epsilon)\lceil \log_2 n \rceil$ times that of the minimum-c-cost of any spanning tree with diameter at most D under d.

The above theorem extends easily to Steiner trees but we omit the details. We shall reserve the term "diameter cost" of a tree to mean the diameter of the tree under the d-costs, and the "building cost" of a tree to mean the total cost of all the edges in the tree computed using the c-costs. We shall also use the term "diameter-D path (tree)" to refer to a path (tree) of diameter cost at most D under d.

We now describe some background material that will be useful in understanding our algorithm and its analysis. Given a diameter bound D, the problem of finding a diameter-D path between two specified vertices of minimum building cost has been termed the multi-objective shortest path problem (MOSP). This problem is NP-complete and Warburton [32] presented the first fully polynomial approximation scheme (FPAS) for this problem. Hassin [19] provided an alternative FPAS for the problem without a running-time dependency on the magnitude of the costs in the problem. His algorithm runs in time $O(|E|(\frac{n^2}{\epsilon} \log \frac{n}{\epsilon}))$, where E is the edge set of the input graph. We use the latter algorithm in implementing our algorithm. Next, we now recall a tree decomposition result used in [22, 28].

Claim 3.7 Let T be a tree with an even number of marked nodes. Then there is a pairing (v_1, w_1), ..., (v_k, w_k) of the marked nodes such that the $v_i - w_i$ paths in T are edge-disjoint.

A pairing of the marked nodes that minimizes the sum of the sizes of the tree-paths between the nodes paired up can be seen to obey the property in the claim above. We use this result in the proof of the performance guarantee.

Overview of the approximation algorithm

The algorithm begins with an empty solution subgraph where each node is in a connected component (termed a *cluster*) by itself in the solution. Assume for simplicity that the number of nodes in the graph n is a power of two. The algorithm works in $\lceil \log_2 n \rceil$ iterations where n is the number of nodes

in the original graph, merging clusters in pairs during each iteration by adding edges between them. This pairing ensures that the number of iterations is as desired.

The clusters maintained by our algorithm represent node-subsets of the input graph G. However, they *do not* represent a partition of the nodes of the input graph. This is because of the way in which we merge the clusters in the algorithm. For each cluster we maintain a spanning tree on the nodes in the cluster. The spanning trees of the clusters maintained by the algorithm are not necessarily edge-disjoint as a result of our merging procedure. We sketch this procedure below. We identify a *center* in the spanning tree of each cluster. In each iteration, every cluster is paired with another cluster and merged with it by the addition of a path between their respective centers. This path may involve nodes that occur in either of the merging clusters or even nodes in other clusters currently maintained by the algorithm. However, while merging two clusters into one, we ensure that the new cluster formed has at most one copy of any node or edge.

The Algorithm

1 Initialize the set of clusters $\mathcal{C}$ to contain n singleton sets, one for each node of the input graph. For each cluster in $\mathcal{C}$, define the single node in the cluster to be the center for the cluster. Initialize the iteration count $i := 1$.

2 Repeat until there remains a single cluster in $\mathcal{C}$

3 Let the set of clusters $\mathcal{C} = \{C_1 \ldots, C_{\frac{n}{2^{i-1}}}\}$.

4 Construct a complete graph G_i as follows.

5 The node set V_i of G_i is $\{v : v \text{ is the center of a cluster in } \mathcal{C}\}$.

6 Between every pair of nodes v_x and v_y in V_i, include an edge (v_x, v_y) in G_i of cost equal to a $(1+\epsilon)$-approximation of the shortest building cost of a diameter-D path between v_x and v_y in G, where ϵ is the accuracy parameter input to the algorithm. Since a FPAS is available to compute such an estimate [19], the costs of all the edges in G_i can be computed in polynomial-time.

7 Find a minimum-cost perfect matching in G_i.

8 For each edge $e = (v_r, v_s)$ in the matching

9 Let P_{rs} be the path in G represented by $e = (v_r, v_s)$. Add this path to merge the clusters C_r and C_s for which v_r and v_s were centers respectively, to form a new cluster C_{rs}, say. The node set of the cluster C_{rs} is defined as the union of the node sets C_r, C_s and the nodes in P_{rs}.

10 Define the union of the edge sets of the spanning trees for C_r and C_s and the set of edges in P_{rs} to be E_{rs}. The edges in E_{rs} form a connected graph on C_{rs}. Choose one of v_r or v_s as the center v_{rs} of the cluster C_{rs}. Using only the diameter cost function d, find a shortest-path tree rooted at v_{rs} in the graph (C_{rs}, E_{rs}). This is the spanning tree for the cluster C_{rs}.

11 Set $\mathcal{C} := \mathcal{C} - \{C_r, C_s\} \cup \{C_{rs}\}$.

12 $i := i + 1$.

13 Output the spanning tree of the single cluster in $\mathcal{C}$.

We prove the performance guarantee using a series of lemmas.

At each iteration, since the clusters are paired up using a perfect matching and merged to form new clusters, the number of clusters halves. Thus we have the following lemma.

Lemma 3.8 The total number of iterations of the above algorithm is $\lceil \log_2 n \rceil$.

In the next lemma, we show that the diameter cost only increases by an additive $2D$ factor at each iteration when we merge two clusters.

Lemma 3.9 Let C be a cluster formed at iteration i of the algorithm. Then the diameter cost of the spanning tree of C maintained by the algorithm is at most $2iD$.

Proof: We prove the lemma by maintaining the following stronger claim inductively.

Claim 3.10 Let C be a cluster with center v formed at iteration i of the algorithm. Then any node u in C has a diameter-iD path to v in the spanning tree of C maintained by the algorithm.

Proof of claim: The proof is by induction on the iteration count i. The basis when $i = 1$ is trivial. To prove the induction step, consider a cluster C_{rs} formed at iteration i (> 1) by merging two clusters C_r and C_s with centers v_r and v_s respectively. Suppose $v_{rs} = v_r$. Consider a node $u \in C_{rs}$. u is in either C_r,C_s or the path P_{rs}. In all these cases, using the inductive hypothesis and the fact that the diameter cost of path P_{rs} is at most D, it is easy to show that u has a diameter-iD path to the center $v_{rs} = v_r$ in the graph (C_{rs}, E_{rs}). Since we compute a shortest-path tree in this graph rooted at v_{rs} using the diameter costs, it follows that the path in this tree between any node and v_{rs} has diameter cost no more than iD. This completes the proof of both the Claim and the Lemma. □

We have the following corollary from the above two lemmas.

Corollary 3.11 The diameter cost of the spanning tree output by the above algorithm is at most $2\lceil \log_2 n \rceil D$.

Lemma 3.12 Let OPT_D be the minimum building cost of any diameter-D spanning tree of the input graph. At each iteration i of the algorithm, the cost of the minimum cost matching in G_i found in Step 7 is at most $(1+\epsilon) \cdot OPT_D$, where $\epsilon > 0$ is the accuracy parameter input to the algorithm.

Proof: It is easy to show using a simple induction on the iteration count that, at any iteration i, the set of centers of clusters for this iteration are distinct nodes of G. Since these are exactly the nodes in G_i, we have that the graph G_i has at most one copy of any node of G. We can now apply Claim 3.7 on the optimal diameter-D spanning tree of the input graph with the nodes of G_i marked. The claim yields a pairing between these centers such that the pairs are connected using edge-disjoint paths in the optimal tree. Note that all these paths have diameter cost at most D since they are derived from a diameter-D tree. Furthermore, the sum of the building costs of all these paths is at most OPT_D since they form edge-disjoint fragments of a tree of total building cost OPT_D. Thus we have identified a pairing between the nodes in G_i of "cost" at most OPT_D where the cost of a pair is the minimum building cost of a diameter-D path between its endpoints.

In constructing G_i, the cost assigned to an edge between a pair of nodes is a $(1+\epsilon)$-approximation to the minimum building cost of a diameter-D path between these nodes in G (see Step 6). Thus between every pair identified above, there is an edge in G_i of cost at most $(1+\epsilon)$ times the building cost of the path between them in the optimal tree. This identifies a perfect matching in G_i of cost at most $(1+\epsilon) \cdot OPT_D$ and completes the proof. □

Note that the spanning tree finally output by the above algorithm is a subgraph of the union of all the paths added by all the matchings over all the iterations. Since the number of iterations is bounded as in Lemma 3.8 and the total building cost of all the edges added in any iteration is bounded as in Lemma 3.12, we have the following bound on the total building cost of the output tree.

Corollary 3.13 The building cost of the spanning tree output by the above algorithm is at most $(1+\epsilon)\lceil \log_2 n \rceil OPT_D$, where $\epsilon > 0$ is the accuracy parameter input to the algorithm.

Corollaries 3.11 and 3.13 with the observation that the algorithm described above runs in polynomial-time together prove Theorem 3.6.

4 Treewidth-bounded Graphs

In this section we consider the class of treewidth-bounded graphs and give algorithms with improved time bounds and superior performance guarantees for several of the bicriteria problems mentioned earlier. We do this in two steps. First we develop pseudopolynomial-time algorithms based on dynamic programming. We then present a general method for deriving fully polynomial-time approximation schemes *(FPAS)* from the pseudopolynomial-time algorithms.

A class of treewidth-bounded graphs can be specified using a finite number of primitive graphs and a finite collection of binary composition rules. We use this characterization for proving our results. A class of treewidth-bounded graphs Γ is inductively defined as follows [9].

1. The number of primitive graphs in Γ is finite.
2. Each graph in Γ has an ordered set of special nodes called **terminals**. The number of terminals in each graph is bounded by a constant.
3. There is a finite collection of binary composition rules that operate only at terminals, either by identifying two terminals or adding an edge between terminals. A composition rule also determines the terminals of the resulting graph, which must be a subset of the terminals of the two graphs being composed.

4.1 Exact algorithms

Theorem 4.1 Given a class of treewidth bounded graphs with no more than k terminals (where k is a constant) and a bound B, each problem Π in Table 2, has an $O((n \cdot B \cdot C)^{O(1)})$-time algorithm for solving the problem Π. Here C denotes an upper bound on the value of the second objective and B denotes the bound on the first objective function.

The above theorem states that there exist pseudopolynomial-time algorithms for all the bicriteria problems from Table 2 when restricted to the class of treewidth-bounded graphs. In fact, the above theorem can be used to obtain a polynomial-time (not just pseudopolynomial-time) algorithm for the degree-bounded minimum cost spanning tree problem since the bounds on the degree of a node in a tree is bounded by n. The basic idea behind all of these algorithms is to employ a dynamic programming strategy. We illustrate this strategy by presenting in some detail the algorithm for the diameter-bounded minimum cost spanning tree problem.

Theorem 4.2 For any class of treewidth-bounded graphs with no more than k terminals, there is an $O(n \cdot k^{2k+4} \cdot B^{O(k^4)})$-time algorithm for solving the diameter B-bounded minimum cost spanning tree problem.

Let f be the cost function on the edges for the first objective (diameter) and g, the cost function for the second objective (total cost). Let Γ be any class of decomposable graphs. Let the maximum number of terminals associated with any graph G in Γ be k. Following [9], it is assumed that a given graph G is accompanied by a parse tree specifying how G is constructed using the rules and that the size of the parse tree is linear in the number of nodes.

Let π be a partition of the terminals of G. For every terminal i let d_i be a number in $\{1 \ldots B\}$. For every pair of terminals i and j in the same block of the partition π let d_{ij} be a number in $\{1 \ldots B\}$. Corresponding to every partition π, set $\{d_i\}$ and set $\{d_{ij}\}$ we associate a cost for G, $Cost^{\pi}_{\{d_i\},\{d_{ij}\}}$ = Minimum total cost under the g function of any forest containing a tree for each block of π, such that the terminal nodes occurring in each tree are exactly the members of the corresponding block of π, no pair of trees is connected, every vertex in G appears in exactly one tree,

d_i is an upper bound on the maximum distance (under the f function) from i to any vertex in the same tree and d_{ij} is an upper bound the distance (under the f function) between terminals i and j in their tree. For the above defined cost, if there is no forest satisfying the required conditions the value of Cost is defined to be ∞.

Note that the number of cost values associated with any graph in Γ is $O(k^k \cdot B^{O(k^2)})$. We now show how the cost values can be computed in a bottom-up manner given the parse tree for G. To begin with, since Γ is fixed, the number of primitive graphs is finite. For a primitive graph, each cost value can be computed in constant time, since the number of forests to be examined is fixed. Now consider computing the cost values for a graph G constructed from subgraphs G_1 and G_2, where the cost values for G_1 and G_2 have already been computed. Notice that any forest realizing a particular cost value for G decomposes into two forests, one for G_1 and one for G_2 with some cost values. Since we have maintained the best cost values for all possibilities for G_1 and G_2, we can reconstruct for each partition of the terminals of G the forest that has minimum cost value among all the forests for this partition obeying the diameter constraints. We can do this in time independent of the sizes of G_1 and G_2 because they interact only at the terminals to form G, and we have maintained all relevant information.

Hence we can generate all possible cost values for G by considering combinations of all relevant pairs of cost values for G_1 and G_2. This takes time $O(k^4)$ per combination for a total time of $O(k^{2k+4} \cdot B^{O(k^4)})$. As in [9], we assume that the size of the given parse tree for G is $O(n)$. Then the dynamic programming algorithm takes time $O(n \cdot k^{2k+4} \cdot B^{O(k^4)})$. This completes the proof of Theorem 4.2.

4.2 Devising polynomial-time approximation schemes

The pseudopolynomial-time algorithms described in the previous section can be used to design fully polynomial-time approximation schemes (FPAS) for these problems for the class of treewidth-bounded graphs. The main difficulty that arises in designing such FPAS is that no trivial upper and lower bounds are known such that their ratio is bounded by a polynomial function of the input. However, we use a technique similar to that of Hassin [19]. We illustrate our ideas once again by devising a FPAS for the (Diameter, Total cost) problem for the class of treewidth-bounded graphs. We now describe our algorithm.

Algorithm DMST-FPAS(ϵ, f, g, B)

Input: A treewidth-bounded graph $G(V, E)$ with two edge weight functions f and g, a bound B and $\epsilon > 0$,

Ouptut: A tree T spanning the nodes of V such that the diameter of tree under g is at most B and the cost of the tree under f is at most $(1+\epsilon)OPT$, where OPT denotes the best tree under f which obeys the diameter bound under g.

1 Choose the initial bounding values UB and LB for the cost of the tree under the f costs as follows: The lower bound LB is the cost of a minimum spanning tree of G under f. The upper bound UB is the sum of the costs of the $n-1$ largest edge costs under f.

2 **While** UB ≥ 2 LB **do**

3 Let $V := (LB \cdot UB)^{1/2}$

4 If Test(V) = YES set LB := V, else

5 Test(V) = NO, then set $UB := V(1+\epsilon)$

6 With the f-costs of edges scaled by a factor of $\frac{(n-1)}{\epsilon LB}$ find a minimum cost diameter-B-bounded tree using dynamic programming. g

Procedure Test(V)

1 $C := 1$; $flag := FALSE$

2 For all edges (i,j) of cost $f_{i,j}$ (under f) do

3 If $f_{i,j} > V$ then throw the edge out **else**

4 $f_{i,j} = \lfloor \frac{f_{ij}}{V\epsilon/(n-1)} \rfloor$

5 **While** $F < \frac{n-1}{\epsilon}$ **and** $flag = FALSE$ **do**

6 If the dynamic programming algorithm for F-total cost-bounded minimum diameter (under g) problem for G outputs a value less than B then $flag = TRUE$.

7 $F := F + 1$.

8 **If** $flag = TRUE$ **then** output NO **else** output YES.

We show that given an $\epsilon > 0$, Procedure **Test** is an ϵ-approximate test. Let V be a given value and assume that we want to test if $OPT \geq V$.

Lemma 4.3 If TEST(V) outputs YES then $OPT > V$; If TEST(V) outputs NO then $OPT < V(1+\epsilon)$.

Proof: Observe that the scaling step decreases the f-cost of each edge f_{ij} by at most $\frac{V\epsilon}{n-1}$ and the total f cost of the tree by at most $V\epsilon$. If the TEST answers NO then we have a tree which has a cost no more than $\frac{V\epsilon}{n-1}F' + V\epsilon < V(1+\epsilon)$, since $F' < \frac{n-1}{\epsilon}$. In the TEST procedure answers YES, then every tree which obeys the delay bound has a scaled cost of at least $F' \geq \frac{(n-1)}{\epsilon}$, which means the unscaled f-cost of any tree is at least V. □

Following [19], the total time required to scale the edges is $O(|E|log(\frac{n}{\epsilon}))$. Executing the dynamic programming algorithm requires $O(\frac{n}{\epsilon}poly(\frac{n}{\epsilon}))$ (the dynamic programming procedure with the total cost bound F is used $\frac{n}{\epsilon}$ times). Observe that the maximum value of the bound placed on the total cost is $O(\frac{n}{\epsilon})$ which is a polynomial in ϵ and the input size. Hence our dynamic programming procedure runs in polynomial-time for any given value of $\epsilon > 0$.

To approximate OPT we now repeat the Test procedure by successively reducing the range between the bounds (LB,UB) in Algorithm **DMST-FPAS** and finally when the ratio is less than two, use a dynamic programming algorithm to derive an ϵ-approximation for the problem. As in [19], our algorithm reduces the ratio $\frac{UB}{LB}$ by performing binary search over the range (LB,UB) in the logarithmic scale. The number of tests to reduce the ratio below two is $O(\log\log(UB/LB))$. The details are identical to [19] and omitted. [5]

Theorem 4.4 For the class of treewidth-bounded graphs, there is a polynomial-time approximation algorithm for the (diameter,cost) problem with performance guarantee $(1, 1+\epsilon)$.

[5] The idea of binary search in the logarithmic scale can be used to replace the ordinary binary search in **Algorithm Two-cost** as well as in the proof of Theorem 3.1 if we are willing to settle for a constant factor increase in the performance guarantees. The details follow from the treatment in [19].

Acknowledgements We thank Professors S. Arnborg and H. L. Bodlaender for pointing out to us the equivalence between treewidth bounded graphs and decomposable graphs. We wish to thank A. Ramesh (Computer Science Department, SUNY Albany) for bringing [25] to our attention. We also thank Dr.Vachaspati Kompella for making his other papers available to us.

References

[1] A.A. Ageev, "A Criterion of Polynomial-Time Solvability for the Network Location Problems," *Proceedings of the third MPS conference on Integer Programming and Combinatorial Optimization Conference* (1993), pp. 237-245.

[2] S. Arnborg, "Efficient Algorithms for Combinatorial Problems on Graphs with Bounded Decomposability," *BIT*, Vol. 25, 1985, pp. 2-23.

[3] S. Arnborg, A. Proskurowski, "Linear-Time Algorithms for NP-hard problems on Graphs Embedded in k-Trees," *Discrete Applied Math.*, Vol. 23, 1989, pp. 11-24.

[4] S. Arnborg, J. Lagergren and D. Seese, "Easy Problems for Tree-Decomposable Graphs," *J. Algorithms*, Vol. 12, 1991, pp. 308-340.

[5] S. Arnborg, B. Courcelle, A. Proskurowski and D. Seese, "An Algebraic Theory of Graph Problems," *J. ACM*, Vol. 12, 1993, pp. 308-340.

[6] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala, "Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen," Technical Report CMU-CS-94-173, (August 1994).

[7] B. Awerbuch, A. Baratz, and D. Peleg, "Cost-sensitive analysis of communication protocols," *Proc. of 9th Symp. on Principles of Distributed Computing (PODC)*, pp. 177-187, (1990).

[8] M. Bellare, S. Goldwasser, C. Lund, and A. Russell, "Efficient probabilistically checkable proofs," *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing* (1993), pp. 294-304.

[9] M.W. Bern, E.L. Lawler and A.L. Wong, "Linear -Time Computation of Optimal Subgraphs of Decomposable Graphs," *J. Algorithms*, Vol. 8, 1987, pp. 216-235.

[10] H.L. Bodlaender, "Dynamic programming on graphs of bounded treewidth," *Proceedings of the 15th ICALP*, LNCS Vol. 317, 1988, pp. 105-118.

[11] H.L. Bodlaender "Polynomial Algorithms for Graph Isomorphism and Chromatic Index on Partial k-Trees," *J. Algorithms*, Vol. 11, 1990, pp. 631-643.

[12] H.L. Bodlaender "A Tourist Guide through Treewidth," *RUU-CS-92-12*, Utrecht University, 1992.

[13] P. M. Camerini, and G. Galbiati, "The bounded path problem," *SIAM J. Alg. Disc., Meth.* Vol. 3, No. 4 (1982), pp. 474-484.

[14] D. Chhajed and T. Lowe, "Solving Structured Multifacility Location Problems Efficiently," *Transportation Science,* Vol. 28, No. 2, May 1994, pp 104-115.

[15] C.-H. Chow, "On multicast path finding algorithms," *Proc. of IEEE INFOCOM '91*, pp. 1274-1283 (1991).

[16] A. Frank, L. Wittie, and A. Bernstein, "Multicast communication in network computers," *IEEE Software*, Vol. 2, No. 3, pp. 49-61 (1985).

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco (1979).

[18] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, E. Tardos and D. P. Williamson, "Improved approximation algorithms for network design problems," *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms,* pp 223-232, (1994).

[19] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Math. of O. R.*, Vol. 17, No. 1, pp. 36-42 (1992).

[20] B. Kadaba and J. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Trans. on Comm.*, Vol. COM-31, pp. 343-351, (Mar. 1983).

[21] S. Khuller, B. Raghavachari, and N. Young, "Balancing Minimum Spanning and Shortest Path Trees," *Proc., Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (1993), pp. 243-250.

[22] P. N. Klein, and R. Ravi, "A nearly best-possible approximation for node-weighted Steiner trees," *Proceedings of the third MPS conference on Integer Programming and Combinatorial Optimization Conference* (1993), pp. 323-332.

[23] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Multicasting for multimedia applications," *Proc. of IEEE INFOCOM '92*, (May 1992).

[24] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Two distributed algorithms for the constrained Steiner tree problem," Technical Report CAL-1005-92, Computer Systems Laboratory, University of California, San Diego, (Oct. 1992).

[25] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Transactions on Networking*, pp. 286-292, (1993).

[26] G. Kortsarz and D. Peleg, "Approximation algorithms for minimum time broadcast," *Proceedings of the 1992 Israel Symposium on Theoretical Computer Science* LNCS 601, (1994).

[27] C. Lund and M. Yannakakis, "On the Hardness of Approximating Minimization Problems," *Proc., 25th Annual ACM Symp. on Theory of Computing*, (1993), pp. 286-293.

[28] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H.B. Hunt III, "Many birds with one stone: Multi-objective approximation algorithms," *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing* (1993), pp. 438-447. (Expanded version appears as Brown University Technical Report TR-CS-92-58.)

[29] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, "Spanning trees short or small," in *Proceedings, Fifth Annual ACM-SIAM Symposium on Discrete Algorithms,* (1994), pp 546-555.

[30] R. Ravi, "Rapid Rumor Ramification: Approximating the minimum broadcast time," to appear in *Proceedings of the 25th Annual IEEE Foundations of Computer Science* (1994).

[31] N. Robertson and P. Seymour, "Graph Minors XIV, Wagners conjecture," *Preprint*, 1992.

[32] A. Warburton, "Approximation of Pareto optima in multiple-objective, shortest path problems," *Oper. Res.* 35, pp. 70-79 (1987).

The Appendix

Proof of Theorem 3.5

We reduce an instance of the set cover problem to a diameter-bounded Steiner tree problem such that any Steiner tree obeying the diameter bound provides a feasible solution for the set cover problem.

An instance of the set cover problem consists of a set G of ground elements $\{g_1, g_2, \ldots, g_k\}$, a collection of subsets $S_1, S_2, \ldots, S_l$ of G where each set S_i has an associated cost c_i. The set cover problem is that of finding a minimum cost collection of the subsets whose union is G.

We construct an instance of the bounded-diameter Steiner tree problem from the set cover problem as follows. The underlying graph has a node g_j for each ground element, a node v_i for each set S_i, and an extra "enforcer-node" v_e. For each set S_i, we add an edge from v_e to v_i of building cost c_i. We also add edges of zero building cost from v_i to all the ground element nodes such that these elements are in S_i. In addition to these edges, we add a path P_e made of two edges of zero building cost to the enforcer v_e. All the edges described above have unit diameter cost. All other edges in the graph are assigned infinite building and diameter costs. We specify the terminals for the Steiner problem as the set of nodes in G along with v_e and the two nodes in P_e.

We claim that any diameter-four Steiner tree provides a feasible cover for the corresponding set cover problem. To show this, we show that in any diameter-four Steiner tree, the path from any ground element node to v_e must use an edge (v_e, v_i) corresponding to a set S_i that contains this element. To see this, observe that as a result of the way we assigned building and diameter costs to all the edges incident on P_e, any diameter-four Steiner tree must contain the whole path P_e. Consider the path from the terminal node of this path to a ground element node. This must pass through v_e and the subpath from v_e to the ground element node has diameter cost at most two. But any path of two edges from v_e to any ground element must use an edge (v_e, v_i) corresponding to a set S_i that contains this element.

Theorem 3.5 then follows from previous results [8, 27] that the set cover problem instance with k ground elements cannot be approximated to better than a factor $\log k/8$ unless $NP \subseteq DTIME(n^{\log\log n})$.

Proof of Theorem 3.2

Reduction from the Partition problem. An instance I of the Partition problem consists of a set $A = \{a_1, a_2, \cdots, a_n\}$ along with weights $s(a_i)$ for each a_i. The problem is whether there exists a set $A_1 \subseteq A$ such that $\sum_{a\in A_1} s(a) = \sum_{a\in A-A_1} s(a)$. Let $\sum_{a\in A} s(a) = 2B$. Given an instance I of the partition problem we construct a series parallel graph with $n+1$ vertices $x_1, x_2, \cdots x_{n+1}$ and $2n$ edges. Between every pair of consecutive vertices x_i and x_{i+1}, $1 \leq i \leq n$, there are two parallel edges $e^1_{i,i+1}$ and $e^2_{i,i+1}$. Now we specify the f-costs and the g-costs for each edge. We will use the notation (a, b) to mean that the f-cost of the edge is a and the g-cost of the edge is b. The cost on edge $e^1_{i,i+1}$ is $(s(a_i), 0)$ and the cost on the edge $e^2_{i,i+1}$ is $(0, s(a_i))$. It can now be easily verified that the graph G has a (Diameter, Total cost) tree of cost (B, B) if and only if the instance of partition problem has a solution. A similar proof yields the NP-hardness proofs for the (Total cost, Total cost) and (Diameter, Diameter) problems. □

Near optimal broadcast schemes

The algorithm for the (Degree, Diameter) case for treewidth-bounded graphs has a nice application. It can be used in conjunction with the ideas presented in [30] to obtain a near-optimal broadcast tree for the class of treewidth-bounded graphs. As mentioned earlier, these results generalize and improve the results of Kortsarz and Peleg.

Given an unweighted graph G and a root r, a broadcast scheme for the graph from the root is one using which a message from the root can be transmitted to all the nodes of G. We consider a telephone model in which the messages are transmitted synchronously and at each time step, any node can either transmit or receive a message from one of its neighbors. The minimum broadcast time problem is to compute a scheme that completes in the minimum number of time steps. Let $OPT_r(G)$ denote the minimum broadcast time for the root and let $OPT(G) = Min_{r \in G} OPT_r(G)$ denote the minimum broadcast time for the graph from any root. We call the problem of computing $OPT_r(G)$ the *minimum rooted broadcast time problem* and that of computing $OPT(G)$ the *minimum broadcast time problem.* These problems are known to be NP-complete [17].

Given a tree T, let $maxdegree_T$ denote the maximum degree of a node in T and let $diameter_T$ denotes the diameter of the tree T. We recall the following definition from [30].

Definition: The poise $P(G)$ of the graph G is defined as follows.

$$P(G) = \min_{\text{spanning trees } T} \{maxdegree_T + diameter_T\}$$

The following theorem [30] points out the relation between the poise of a graph and the optimal broadcast scheme.

Theorem 4.5

$$\Omega(P(G)) \leq OPT(G) \leq O(OPT(G) \cdot P(G) \cdot \frac{\log n}{\log \log n})$$

The proof of this theorem is constructive and implies that a good solution to the poise of a graph can be used to construct a good solution for the minimum broadcast time problem with a factor of $O(\frac{\log n}{\log \log n})$ overhead. Using our results for the (Degree, Diameter) problem for treewidth bounded graphs, we thus obtain the following.

Theorem 4.6 For the class of treewidth bounded graphs, there is a polynomial-time algorithm to compute a spanning tree T such that $(maxdegree_T + diameter_T) = P(G)$, where $P(G)$ denotes the poise of the graph G.

Proof Sketch : Use the (Degree, Diameter) approximation algorithm to compute degree-bounded minimum diameter spanning trees for all possible degree bounds. Since the graph is unweighted, the overall running time is strongly polynomial. Choose the tree T with the least value of $(degree_T + diameter_T)$. □

¿From Theorems 4.6 and 4.5 we get the following theorem

Theorem 4.7 For any class of treewidth-bounded graphs there is a polynomial-time $O(\frac{\log n}{\log \log n})$-approximation algorithm for the minimum rooted broadcast time problem.