Chapter 12

Bicriteria Network Design problems

12.1 Introduction

A generic bicriteria network design problem, (**A**, **B**, **S**), is defined by identifying two minimization objectives, **A** and **B**, from a set of possible objectives, and specifying a membership requirement in a class of subgraphs, **S**. The problem specifies a budget value on the first objective, **A**, under one cost function, and seeks to find a network having minimum possible value for the second objective, **B**, under another cost function, such that this network is within the budget on the first objective. The solution network must belong to the subgraph-class **S**.

As an example, consider the problem of designing networks capable of accommodating multimedia (both audio and video) traffic in a multicast (simultaneous transmission of data to multiple destinations) environment [Ch91, FW+85, KJ83, KP+92A, KP+92B, KP+93]. As argued in [KP+92A], one of the popular solutions to multicast routing involves tree construction. Two optimization criteria – (1) the minimum worst-case transmission delay and (2) the minimum total cost – are typically sought to be minimized in the construction of these trees. As pointed out in [KP+92A], in the problem of finding good multicast trees, each edge has associated with it two edge costs: the construction cost and the delay cost. The construction cost is typically a measure of the amount of buffer space or channel bandwidth used and the delay cost is a combination of the propagation, transmission and queuing delays.

In our terminology, the problem of finding low-cost and low-transmission-delay multimedia networks [KP+92A, KP+93] can be modeled as the (Diameter, Total cost, Spanning tree)-bicriteria problem: given an undirected graph $G = (V, E)$ with two different weight functions c_e (modeling the construction cost) and d_e (modeling the delay cost) for each edge $e \in E$, and a bound $\mathcal{D}$ (on the total delay), find a minimum c-cost spanning tree such that the diameter of the tree under the d-costs is at most $\mathcal{D}$.

Such multi-criteria network design problems, with separate cost functions for each optimization criterion, also occur naturally in VLSI designs (see [ZP+94] and the references therein). With the advent of deep micron VLSI designs, the feature size has shrunk to sizes of 0.5 microns and less. As a result, the interconnect resistance, being proportional to the square of the scaling factor, has increased significantly. An increase in interconnect resistance has led to an increase in interconnect delays thus making them a dominant factor in the timing analysis of VLSI circuits. Therefore

VLSI circuit designers aim at finding minimum cost (spanning or Steiner) trees given delay bound constraints on source-sink connections.

Network design problems where even one cost measure must be minimized, are often **NP**-hard [GJ79]. But, in several applications, it is often the case that the network to be built is required to minimize multiple cost measures simultaneously, with different cost functions for each measure. In the past, the problem of minimizing two cost measures was often dealt with by attempting to minimize some combination of the two, thus converting it into a uni-criterion problem. This approach often fails, especially when the two criteria are very disparate. We have chosen, instead, to model bicriteria problems as that of minimizing one criterion subject to a budget on the other. We argue that this approach is both general as well as robust. It is more general because it subsumes the case where one wishes to minimize some functional combination of the two criteria. It is more robust because the quality of approximation is independent of which of the two criteria we impose the budget on. We elaborate on this more in Sections 12.3.1 and 12.3.2.

12.1.1 Objective functions

In this class, we study the complexity and approximability of a number of bicriteria network design problems. The three objectives we consider are: (i) total cost, (ii) diameter and (iii) degree of the network. These reflect the price of synthesizing the network, the maximum delay between two points in the network and the reliability of the network, respectively. The *Total cost* objective is the sum of the costs of all the edges in the subgraph. The *Diameter* objective is the maximum distance between any pair of nodes in the subgraph. The *Degree* objective denotes the maximum over all nodes in the subgraph, of the degree of the node. We may generalize the degree objective to the case of *Weighted Degree* given costs on the edges as follows: the weighted degree of a node in a subgraph is the sum of the costs of the edges incident on the node in the subgraph. The weighted degree objective is the maximum over all nodes in the subgraph, of the weighted degree of the node. When all the edges have unit costs, the weighted degree objective reduces to the regular degree objective.

The class of subgraphs we consider in are mainly one-connected networks such as *Spanning trees*. However, most of the results we discuss also extend to more general one-connected networks such as Steiner trees and generalized Steiner forests[AK+95, GW92].

12.1.2 Performance guarantees

As we mentioned earlier, most of the problems considered in this paper, are **NP**-hard for arbitrary instances even when we wish to find optimum solutions with respect to a single criterion. Given the hardness of finding optimal solutions, we concentrate on devising approximation algorithms with worst case performance guarantees. Recall that an approximation algorithm for a minimization problem Π provides a **performance guarantee** of ρ if for every instance I of Π, the solution value returned by the approximation algorithm is within a factor ρ of the optimal value for I. Here, we extend this notion to apply to bicriteria optimization problems. An (α, β)-approximation algorithm for an (**A**, **B**, **S**)-bicriteria problem is defined as a polynomial-time algorithm that produces a solution in which the first objective (**A**) value, is at most α times the budget, and the second objective (**B**) value, is at most β times the minimum for any solution that is within the budget on **A**. The solution produced must belong to the subgraph-class **S**. Analogous definitions can be given when **A** and/or **B** are maximization objectives.

The main results we discuss in this note are as follows.

1. Very often, bicriteria problems are harder than uni-criterion problems even to approximate. We demonstrate this by showing an approximation-preserving reduction from the Set Cover problem to the (Diameter, Total cost, Spanning tree) problem that proves that obtaining a $(< \frac{7}{6}, < \ln n)$ approximation for the latter problem is NP-hard. Such "double" hardness results also justify the definition of bicriteria approximations in the way we did.
2. We justify the claim that the formulation of bicriteria problem by using a budget for one objective and minimizing the other is more robust than alternative formulations, and remains consistent irrespective of which of the two objectives we choose to budget.
3. For bicriteria problems where both the objectives are similar, i.e., represent the same objective function but computed under two distinct cost functions (e.g. (Total cost, Total cost, Spanning tree)), we show a simple method based on parametric search to convert uni-criterion approximation results for this objective to a bicriteria result.
4. We present a general framework for approximating bicriteria one-connected network problems that is applicable to all pairwise combinations of the three objective functions we mentioned earlier (total cost, diameter and degree) to give performance ratio $(\log n, \log n)$ on an n-node graph. Given the framework, it remains to reason and fill in the appropriate polynomial time subroutine that is applicable for the corresponding pair of objectives.

Table 1 contains the performance guarantees of our approximation algorithms for finding spanning trees, **S**, under different pairs of minimization objectives, **A** and **B**. For each problem cataloged in the table, two different costs are specified on the edges of the undirected graph: the first objective is computed using the first cost function and the second objective, using the second cost function. The rows are indexed by the budgeted objective. For example the entry in row **A**, column **B**, denotes the performance guarantee for the problem of minimizing objective **B** with a budget on the objective **A**. All the results in Table 1 extend to finding Steiner trees with at most a constant factor worsening in the performance ratios. All the results in the table extend to finding Steiner trees with at most a constant factor worsening in the performance ratios (Exercise!).

Cost Measures	Degree	Diameter	Total Cost
Degree	$(O(\log n), O(\log n))^*$	$(O(\log^2 n), O(\log n))$[Ra94]	$(O(\log n), O(\log n))$[RM+93]
Diameter	$(O(\log n), O(\log^2 n))$[Ra94]	$(1+\gamma, 1+\frac{1}{\gamma})^*$	$(O(\log n), O(\log n))^*$
Total Cost	$(O(\log n), O(\log n))$[RM+93]	$(O(\log n), O(\log n))^*$	$(1+\gamma, 1+\frac{1}{\gamma})^*$

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.

The diagonal entries in the table follow as a corollary of a general result (Theorem 12.8) which is proved using a parametric search algorithm. The entry for (Degree, Degree, Spanning tree) follows by combining Theorem 12.8 with the $O(\log n)$-approximation algorithm for the degree problem in [RM+93]. In [RM+93] they actually provide an $O(\log n)$-approximation algorithm for the weighted degree problem. (The weighted degree of a subgraph is defined as the maximum over all nodes of the sum of the costs of the edges incident on the node in the subgraph). Hence we actually get an $(O(\log n), O(\log n))$-approximation algorithm for the (Weighted degree, Weighted degree, Spanning tree)-bicriteria problem. Similarly, the entry for (Diameter, Diameter, Spanning tree) follows by combining Theorem 12.8 with the known exact algorithms for minimum diameter spanning trees [CG82, RS+94]; while the result for (Total cost, Total cost, Spanning tree) follows by combining Theorem 12.8 with an exact algorithm to compute a minimum spanning tree [Kr56, Pr57].

We also describe a different cluster-based approximation algorithm and a solution based decomposition technique for devising approximation algorithms for problems when the two objectives are different. In this note, we describe this technique techniques yield $(O(\log n), O(\log n))$-approximation algorithms for the (Diameter, Total cost, Steiner tree) and the (Degree, Total cost, Steiner tree) problems.

12.1.3 Previous Work

The area of uni-criterion optimization problems for network design is vast and well-explored (See [Ho95, CK95] and the references therein.). Ravi et al. [RM+93] studied the degree-bounded minimum cost spanning tree problem and provided an approximation algorithm with performance guarantee $(O(\log n), O(\log n))$. Though they were doing bicriteria optimization they did not state it as such in their paper.

The (Degree, Diameter, Spanning tree) problem was studied in [Ra94] in the context of minimizing broadcast time in arbitrary networks. There he provides an approximation algorithm for the (Degree, Diameter, Spanning tree) problem with performance guarantee $(O(\log^2 n), O(\log n))$[1].

The (Diameter, Total cost, Spanning tree) entry in Table 1 corresponds to the diameter-constrained minimum spanning tree problem introduced earlier. It is known that this problem is **NP**-hard even in the special case where the two cost functions are identical [HL+89]. Awerbuch, Baratz and Peleg [AB+90] gave an approximation algorithm with $(O(1), O(1))$ performance guarantee for this problem - i.e. the problem of 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. Khuller, Raghavachari and Young [KR+93] studied an extension called *Light, approximate Shortest-path Trees (LAST)* and gave an approximation algorithm with $(O(1), O(1))$ performance guarantee. Kadaba and Jaffe [KJ83], Kompella et al. [KP+92A], and Zhu et al. [ZP+94] considered the (Diameter, Total cost, Steiner tree) problem with two edge costs and presented heuristics without any guarantees. It is easy to construct examples to show that the solutions produced by these heuristics in [ZP+94, KP+92A], can be arbitrarily bad with respect to an optimal solution. A closely related problem is that of finding a diameter-constrained shortest path between two pre-specified vertices s and t, or (Diameter, Total cost, s-t path). This problem, termed the multi-objective shortest path problem (MOSP) in the literature, is **NP**-complete and Warburton [Wa87] presented the first fully polynomial approximation scheme (**FPAS**) for it. Hassin [Ha92] provided a strongly polynomial **FPAS** for the problem which improved the running time of Warburton [Wa87]. This result was further improved by Philips [Ph+93].

The (Total cost, Total cost, Spanning tree)-bicriteria problem has been recently studied by Ganley et al. [GG+95]. They consider a more general problem with more than two weight functions.

[1] The result in [Ra94] is actually somewhat stronger - given a budget, D, on the degree he finds a tree whose total cost is at most $O(\log n)$ times the optimal and whose degree is at most $O(D \log n + \log^2 n)$.

They also gave approximation algorithms for the restricted case when each weight function obeys triangle inequality. However, their algorithm does not have a bounded performance guarantee with respect to each objective.

12.2 Hardness results

The problem of finding a minimum degree spanning tree is strongly **NP**-hard [GJ79]. This implies that all spanning tree bicriteria problems, where one of the criteria is degree, are also strongly **NP**-hard. In contrast, it is well known that the minimum diameter spanning tree problem and the minimum cost spanning tree problems have polynomial time algorithms [CG82, HL+89, Kr56, Pr57, RS+94].

The (Diameter, Total Cost, Spanning tree)-bicriteria problem is strongly **NP**-hard even in the case where both cost functions are identical [HL+89]. We now show that the (Diameter, Total-cost, Steiner tree) problem is hard to approximate within a logarithmic factor. This is in contrast to the approximation algorithm provided in Section 12.5. There is however a gap between the results of Theorems 12.2 and 12.15.

Our non-approximability result is obtained by an approximation preserving reduction from the **MIN SET COVER**. An instance (T, X) of the **MIN SET COVER** problem consists of a universe $T = \{t_1, t_2, \ldots, t_k\}$ and a collection of subsets $X = \{X_1, X_2, \ldots, X_m\}$, $X_i \subseteq T$, each set X_i having an associated cost c_i. The problem is to find a minimum cost collection of the subsets whose union is T. Recently Feige [Fe95] has shown the following non-approximability result:

Theorem 12.1 *Unless $NP \subseteq DTIME(n^{\log\log n})$, the* **MIN SET COVER** *problem, with a universe of size k, cannot be approximated to better than a $\ln k$ factor.*

Theorem 12.2 *Unless $NP \subseteq DTIME(n^{\log\log n})$, given an instance of the (Diameter, Total 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 $R < \ln k$.*

Proof: Approximation preserving reduction from the **MIN SET COVER** problem to the (Diameter, Total Cost, Steiner tree) problem. Given an instance (T, X) of the **MIN SET COVER** problem where $T = \{t_1, t_2, \ldots, t_k\}$ and $X = \{X_1, X_2, \ldots, X_m\}$, $X_i \subseteq T$, where the cost of the set X_i is c_i, we construct an instance G of the (Diameter, Total Cost, Steiner tree) problem as follows. The graph G has a node t_i for each element t_i of T[2], a node x_i for each set X_i, and an extra "enforcer-node" n. For each set X_i, we attach an edge between nodes n and x_i of c-cost c_i, and d-cost 1. For each element t_i and set X_j such that $t_i \in X_j$ we attach an edge (t_i, x_j) of c-cost, 0, and d-cost, 1. In addition to these edges, we add a path P made of two edges of c-cost, 0, and d-cost, 1, to the enforcer node n All other edges in the graph are assigned infinite c and d-costs. The nodes t_i along with n and the two nodes of P are specified to be the terminals for the Steiner tree problem instance. We claim that G has a c-cost Steiner tree of diameter at most 4 and cost $\mathcal{C}$ if and only if the original instance (T, X) has a solution of cost $\mathcal{C}$.

[2] There is a mild abuse of notation here but it should not lead to any confusion.

It is easy to see that any Steiner tree of diameter at most 4 must contain a path from t_i to n, for all i, that uses an edge (x_j, n) for some X_j such that $t_i \in X_j$. Hence any Steiner tree of diameter at most 4 provides a feasible solution of equivalent c-cost to the original Set cover instance. The proof now follows from Theorem 12.1.

∎

Exercise Show that a construction similar to the one above can be used to strengthen the condition of the output Steiner tree having diameter at most D in the above Theorem to having diameter less than $\frac{6D}{5}$.

12.3 Bicriteria Formulations: Simple Properties

12.3.1 Equivalence of Bicriteria Formulations: Robustness

There are two natural alternative ways of formulating general bicriteria problems: (i) where we impose the budget on the first objective and seek to minimize the second and (ii) 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 our definition of a bicriteria approximation is independent of the choice of the criterion that is budgeted in the formulation. This makes it a robust definition and allows us to fill in the entries for the problems (**B**, **A**, **S**) by transforming the results for the corresponding problems (**A**, **B**, **S**).

Let G be a graph with two (integral)[3] cost functions, c and d. c and d are typically edge costs or node costs. Let **A** (**B**) be a minimization objective computed using cost function c (d). Let the budget bound on the c-cost[4] (d-cost) of a solution subgraph be denoted by $\mathcal{C}$ ($\mathcal{D}$).

There are two natural ways to formulate a bicriteria problem: (i) (**A**, **B**, **S**) - find a subgraph in **S** whose **A**-objective value (under the c-cost) is at most $\mathcal{C}$ and which has minimum **B**-objective value (under the d-cost), (ii) (**B**, **A**, **S**) - find a subgraph in **S** whose **B**-objective value (under the d-cost) is at most $\mathcal{D}$ and which has minimum **A**-objective value (under the c-cost).

Note that bicriteria problems are meaningful only when the two criteria are *hostile* with respect to each other - the minimization of one criterion conflicts with the minimization of the other. A good example of hostile objectives are the degree and the total edge cost of a spanning tree in an unweighted graph [RM+93]. Two minimization criteria are formally defined to be hostile whenever the minimum value of one objective is monotonically nondecreasing as the budget (bound) on the value of the other objective is decreased.

Let $\mathbf{XYZ}(G, \mathcal{C})$ be any (α, β)-approximation algorithm for (**A**, **B**, **S**) on graph G with budget $\mathcal{C}$ under the c-cost. We now show that there is a transformation which produces a (β, α)-approximation algorithm for (**B**, **A**, **S**). The transformation uses binary search on the range of values of the c-cost with an application of the given approximation algorithm, **XYZ**, at each step of this search. Let the minimum c-cost of a $\mathcal{D}$-bounded subgraph in **S** be OPT_c. Let $\mathcal{C}_{hi}$ be an upper

[3] In case of rational cost functions, our algorithms can be easily extended with a small additive loss in the performance guarantee.

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

bound on the c-cost of any $\mathcal{D}$-bounded subgraph in $\mathbf{S}$. Note that $\mathcal{C}_{hi}$ is at most some polynomial in n times the maximum c-cost (of an edge or a node). Hence $\log(\mathcal{C}_{hi})$ is at most a polynomial in terms of the input specification. Let Heu_c (Heu_d) denote the c-cost (d-cost) of the subgraph output by ALGORITHM BICRITERIA-EQUIVALENCE given below.

ALGORITHM BICRITERIA-EQUIVALENCE:

- *Input:* G - graph, $\mathcal{D}$ - budget on criterion $\mathbf{B}$ under the d-cost, $\mathbf{XYZ}$ - an (α, β)-approximation algorithm for ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{S}$).
- 1. Let $\mathcal{C}_{hi}$ be an upper bound on the c-cost of any $\mathcal{D}$-bounded subgraph in $\mathbf{S}$.
  2. Do binary search and find a $\mathcal{C}'$ in $[0, \mathcal{C}_{hi}]$ such that
     (a) $\mathbf{XYZ}(G, \mathcal{C}')$ returns a subgraph with d-cost greater than $\beta\mathcal{D}$, and
     (b) $\mathbf{XYZ}(G, \mathcal{C}'+1)$ returns a subgraph with d-cost at most $\beta\mathcal{D}$.
  3. If the binary search in Step 2 fails to find a valid $\mathcal{C}'$ then output "NO SOLUTION" else output $\mathbf{XYZ}(G, \mathcal{C}'+1)$.
- *Output:* A subgraph from $\mathbf{S}$ such that its c-cost is at most α times that of the minimum c-cost $\mathcal{D}$-bounded subgraph and its d-cost is at most $\beta\mathcal{D}$.

Claim 12.3 *If G contains a $\mathcal{D}$-bounded subgraph in $\mathbf{S}$ then* ALGORITHM BICRITERIA-EQUIVALENCE *outputs a subgraph from $\mathbf{S}$ whose c-cost is at most α times that of the minimum c-cost $\mathcal{D}$-bounded subgraph and whose d-cost is at most $\beta\mathcal{D}$.*

Proof: Since $\mathbf{A}$ and $\mathbf{B}$ are hostile criteria it follows that the binary search in Step 2 is well defined. Assume that $\mathbf{S}$ contains a $\mathcal{D}$-bounded subgraph. Then, since $\mathbf{XYZ}(G, \mathcal{C}_{hi})$ returns a subgraph with d-cost at most $\beta\mathcal{D}$, it is clear that ALGORITHM BICRITERIA-EQUIVALENCE outputs a subgraph in this case. As a consequence of Step 2a and the performance guarantee of the approximation algorithm $\mathbf{XYZ}$, we get that $\mathcal{C}'+1 \leq OPT_c$. By Step 2b we have that $Heu_d \leq \beta\mathcal{D}$ and $Heu_c \leq \alpha(\mathcal{C}'+1) \leq \alpha OPT_c$. Thus ALGORITHM BICRITERIA-EQUIVALENCE outputs a subgraph from $\mathbf{S}$ whose c-cost is at most α times that of the minimum c-cost $\mathcal{D}$-bounded subgraph and whose d-cost is at most $\beta\mathcal{D}$.

∎

Note however that in general the resulting (β, α)-approximation algorithm is, not *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 $\log \mathcal{C}_{hi}$ the largest c-cost. The above discussion leads to the following theorem.

Theorem 12.4 *Any (α, β)-approximation algorithm for ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{S}$) can be transformed in polynomial time into a (β, α)-approximation algorithm for ($\mathbf{B}$, $\mathbf{A}$, $\mathbf{S}$).*

12.3.2 Comparing with other functional combinations: Generality

Our formulation is more general because it subsumes the case where one wishes to minimize some functional combination of the two criteria. We briefly comment on this next. For the purposes of illustration let **A** and **B** be two objective functions and let us say that we wish to minimize the sum of the two objectives **A** and **B**. Call this an (**A** + **B**, **S**) problem. Let $\mathbf{XYZ}(G, \mathcal{C})$ be any (α, β)-approximation algorithm for (**A**, **B**, **S**) on graph G with budget $\mathcal{C}$ under the c-cost. We show that there is a polynomial time $\max\{\alpha, \beta\}$-approximation algorithm for the (**A** + **B**, **S**) problem. The transformation uses binary search on the range of values of the c-cost with an application of the given approximation algorithm, **XYZ**, at each step of this search. Let the optimum value for the (**A** + **B**, **S**) problem on a graph G be $OPT_{c+d} = (V_c + V_d)$, where V_c and V_d denote respectively the contribution of the two costs c and d for **A** and **B**. Let $Heu_c(\mathcal{C})$ ($Heu_d(\mathcal{C})$) denote the c-cost (d-cost) of the subgraph output by $\mathbf{XYZ}(G, \mathcal{C})$. Finally, let $Heu_{c+d}(\mathcal{C})$ denote the value computed by ALGORITHM CONVERT.

ALGORITHM CONVERT:

- *Input:* G - graph, **XYZ** - an (α, β)-approximation algorithm for (**A**, **B**, **S**).
- 1. Let $\mathcal{C}_{hi}$ be an upper bound on the c-cost of any subgraph in **S**.
  2. Employ binary search to find a $\mathcal{C}'$ in $[0, \mathcal{C}_{hi}]$ such that
     (a) $\mathbf{XYZ}(G, \mathcal{C}')$ returns a subgraph with the minimum value of $Heu_c(\mathcal{C}') + Heu_d(\mathcal{C}')$.
- *Output:* A subgraph from **S** such that the sum of its c-cost and its d-costs is at most $\max\{\alpha, \beta\}(OPT_{c+d})$.

Theorem 12.5 *Let $\mathbf{XYZ}(G, \mathcal{C})$ be any (α, β)-approximation algorithm for (**A**, **B**, **S**) on graph G with budget $\mathcal{C}$ under the c-cost. Then, there is a polynomial time $\max\{\alpha, \beta\}$-approximation algorithm for the (**A** + **B**, **S**) problem.*

Proof Sketch: Consider the iteration of the binary search in which the bound on the c-cost is V_c. Then as a consequence of the performance guarantee of the approximation algorithm **XYZ**, we get that $Heu_c(V_c) \leq \alpha V_c$. By Step 2a and the performance guarantee of the algorithm **XYZ**, we have that $Heu_d(V_c) \leq \beta V_d$. Thus $Heu_{c+d}(V_c) \leq \alpha V_c + \beta V_d \leq \max\{\alpha, \beta\}(V_c + V_d)$. Since ALGORITHM CONVERT outputs a subgraph from **S** the sum of whose c-cost and d-cost is minimized, we have that

$$\min_{\mathcal{C}' \in [0, \mathcal{C}_{hi}]} (Heu_c(\mathcal{C}') + Heu_d(\mathcal{C}')) \leq \max\{\alpha, \beta\}(OPT_{c+d}).$$

A similar argument shows that an (α, β)-approximation algorithm $\mathbf{XYZ}(G, \mathcal{C})$, for a (**A**, **B**, **S**) problem can be used to find devise a polynomial time $\alpha\beta$ approximation algorithm for the (**A** $\times$ **B**, **S**) problem. A similar argument can also be given for other basic functional combinations. We make two additional remarks.

1. The use of approximation algorithms for (**A**, **B**, **S**)-bicriteria problems, to solve (f(**A**, **B**), **S**) problems (f denotes a function combination of the objectives) does not always yield the best possible solutions. For example problems such as (Total Cost + Total Cost , Spanning Tree) and (Total Cost/Total Cost , Spanning Tree) [Ch77, Me83] can be solved exactly in

polynomial time by direct methods but can only be solved approximately using any algorithm for the (Total Cost, Total Cost , Spanning Tree)-bicriteria problem.[5]

2. Algorithms for solving (f(**A**, **B**), **S**) problems can not in general guarantee any bounded performance ratios for solving the (**A**, **B**, **S**) problem. For example, a solution for the (Total Cost + Total Cost , Spanning Tree) problem or the (Total Cost/Total Cost , Spanning Tree) problem can not be directly used to find a good (α, β)-approximation algorithm for the (Total Cost, Total Cost, Spanning Tree)-bicriteria problem.

The above discussion points out that while a good solution to the (**A**, **B**, **S**)-bicriteria problem yields a "good" solution to any uni-criterion version, the converse is not true. It is in this sense that we say our formulation of bicriteria network design problems is general and subsumes other functional combinations.

12.4 Parametric Search

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.).

As before, let G be a graph with two (integral) cost functions, c and d. Let $\mathcal{C}$ denote the budget on criteria **A**. We assume that the c and d cost functions are of the same kind, i.e. they are both costs on edges or, costs on nodes. Let $\mathbf{UVW}(G, f)$ be any ρ-approximation algorithm that on input G produces a solution subgraph in **S** minimizing criterion **A**, under the single cost function f. In a mild abuse of notation, we also let $\mathbf{UVW}(G, f)$ denote the (f-)cost of the subgraph output by $\mathbf{UVW}(G, f)$ when running on input G under cost function f. We use the following additional notation in the description of the algorithm and the proof of its performance guarantee. Given constants a and b and two cost functions f and g, defined on edges (nodes) of a graph, $af + bg$ denotes the composite function that assigns a cost $af(e) + bg(e)$ to each edge (node) in the graph. Let $h(\hat{\mathcal{D}})$ denote the cost of the subgraph, returned by $\mathbf{UVW}(G, (\frac{\hat{\mathcal{D}}}{\mathcal{C}})c + d)$ (under the $((\frac{\hat{\mathcal{D}}}{\mathcal{C}})c + d)$-cost function). Let the minimum d-cost of a $\mathcal{C}$-bounded subgraph in **S** be OPT_d. Let Heu_c (Heu_d) denote the c-cost (d-cost) of the subgraph output by Algorithm Parametric-Search given below.

Let $\gamma > 0$ be a fixed accuracy parameter. In what follows, we devise a $((1 + \gamma), (1 + \frac{1}{\gamma}))$-approximation algorithm for (**A**, **A**, **S**), under the two cost functions c and d. The algorithm consists of performing a binary search with an application of the given approximation algorithm, **UVW**, at each step of this search.

[5]This is true since the (Total Cost, Total Cost, Spanning Tree)-bicriteria problem is **NP**-complete and therefore unless **P** = **NP** can not be solved in polynomial time.

ALGORITHM PARAMETRIC-SEARCH:

- *Input:* G - graph, $\mathcal{C}$ - budget on criteria **A** under the c-cost, **UVW** - a ρ-approximation algorithm that produces a solution subgraph in **S** minimizing criterion **A**, under a single cost function, γ - an accuracy parameter.
- 1. Let $\mathcal{D}_{hi}$ be an upper bound on the d-cost of any $\mathcal{C}$-bounded subgraph in **S**.
  2. Do binary search and find a $\mathcal{D}'$ in $[0, \frac{\mathcal{D}_{hi}}{\gamma}]$ such that
     (a) $\mathbf{UVW}(G, (\frac{\mathcal{D}'}{\mathcal{C}})c + d)$ returns a subgraph such that $\frac{h(\mathcal{D}')}{\mathcal{D}'} > (1+\gamma)\rho$, and
     (b) $\mathbf{UVW}(G, (\frac{\mathcal{D}'+1}{\mathcal{C}})c + d)$ returns a subgraph such that $\frac{h(\mathcal{D}'+1)}{(\mathcal{D}'+1)} \leq (1+\gamma)\rho$.
  3. If the binary search in Step 2 fails to find a valid $\mathcal{C}'$ then output "NO SOLUTION" else output $\mathbf{UVW}(G, (\frac{\mathcal{D}'+1}{\mathcal{C}})c + d)$.
- *Output:* A subgraph from **S** such that its d-cost is at most $(1+\frac{1}{\gamma})\rho$ times that of the minimum d-cost $\mathcal{C}$-bounded subgraph and its c-cost is at most $(1+\gamma)\rho\mathcal{C}$.

Claim 12.6 *The binary search, in Step 2 of* ALGORITHM PARAMETRIC-SEARCH *is well-defined.*

Proof: Since $(\frac{1}{R}\mathbf{UVW}(G, f))$ is the same as $\mathbf{UVW}(G, \frac{f}{R})$, we get that $\frac{h(\hat{\mathcal{D}})}{\hat{\mathcal{D}}} = \frac{1}{\hat{\mathcal{D}}}\,\mathbf{UVW}(G, (\frac{\hat{\mathcal{D}}}{\mathcal{C}})c + d) = \mathbf{UVW}(G, (\frac{1}{\mathcal{C}})c + \frac{1}{\hat{\mathcal{D}}}d)$, Hence $\frac{h(\hat{\mathcal{D}})}{\hat{\mathcal{D}}}$ is a monotone non-increasing function of $\hat{D}$. Thus the binary search in Step 2 of ALGORITHM PARAMETRIC-SEARCH is well-defined. ∎

Claim 12.7 *If G contains a $\mathcal{C}$-bounded subgraph in* **S** *then* ALGORITHM PARAMETRIC-SEARCH *outputs a subgraph from* **S** *whose d-cost is at most $(1+\frac{1}{\gamma})\rho$ times that of the minimum d-cost $\mathcal{C}$-bounded subgraph and whose c-cost is at most $(1+\gamma)\rho\mathcal{C}$.*

Proof: By claim 12.6 we have that the binary search in Step 2 of ALGORITHM PARAMETRIC-SEARCH is well-defined.

Assume that **S** contains a $\mathcal{C}$-bounded subgraph. Then, since $\mathbf{UVW}(G, (\frac{\mathcal{D}_{hi}}{\gamma\mathcal{C}})c + d)$ returns a subgraph with cost at most $(1+\gamma)\rho\mathcal{D}_{hi}$, under the $((\frac{\mathcal{D}_{hi}}{\gamma\mathcal{C}})c + d)$-cost function, it is clear that ALGORITHM PARAMETRIC-SEARCH outputs a subgraph in this case.

As a consequence of Step 2a and the performance guarantee of the approximation algorithm **UVW**, we get that

$$\mathcal{D}' + 1 \leq \frac{OPT_d}{\gamma}.$$

By Step 2b we have that the subgraph output by ALGORITHM PARAMETRIC-SEARCH has the following bounds on the c-costs and the d-costs.

$$Heu_d \leq h(\mathcal{D}'+1) \leq \rho(1+\gamma)(\mathcal{D}'+1) \leq (1+\frac{1}{\gamma})\rho OPT_d$$

$$Heu_c \leq (\frac{\mathcal{C}}{\mathcal{D}'+1})h(\mathcal{D}'+1) \leq (\frac{\mathcal{C}}{\mathcal{D}'+1})(1+\gamma)\rho(\mathcal{D}'+1) = (1+\gamma)\rho\mathcal{C}.$$

Thus ALGORITHM PARAMETRIC-SEARCH outputs a subgraph from $\mathbf{S}$ whose d-cost is at most $(1+\frac{1}{\gamma})\rho$ times that of the minimum d-cost $\mathcal{C}$-bounded subgraph and whose c-cost is at most $(1+\gamma)\rho\mathcal{C}$.

∎

Note however that the resulting $((1+\gamma)\rho, (1+\frac{1}{\gamma})\rho)$-approximation algorithm for ($\mathbf{A}$, $\mathbf{A}$, $\mathbf{S}$) may not be *strongly* polynomial since it depends on the range of the d-costs. But it is a *polynomial-time* algorithm since its running time is linearly dependent on $\log D_{hi}$. Note that $\mathcal{D}_{hi}$ is at most some polynomial in n times the maximum d-cost (of an edge or a node). Hence $\log(\mathcal{D}_{hi})$ is at most a polynomial in terms of the input specification.

The above discussion leads to the following theorem.

Theorem 12.8 *Any ρ-approximation algorithm that produces a solution subgraph in* $\mathbf{S}$ *minimizing criterion* $\mathbf{A}$ *can be transformed into a $((1+\gamma)\rho, (1+\frac{1}{\gamma})\rho)$-approximation algorithm for* ($\mathbf{A}$,$\mathbf{A}$,$\mathbf{S}$).

12.5 Diameter-Constrained Trees

In this section, we describe ALGORITHM DCST, our $(O(\log n), O(\log n))$-approximation algorithm for (Diameter, Total cost, Steiner tree) or the diameter-bounded minimum Steiner tree problem. Note that (Diameter, Total cost, Steiner tree) includes (Diameter, Total cost, Spanning tree) as a special case. We first state the problem formally: given an undirected graph $G = (V, E)$, with two cost functions c and d defined on the set of edges, diameter bound D and terminal set $K \subseteq V$, the (Diameter, Total cost, Steiner tree) problem is to find a tree of minimum c-cost connecting the set of terminals in K with diameter at most D under the d-cost.

The technique underlying ALGORITHM DCST is very general and has wide applicability. Hence, we first give a brief synopsis of it. The basic algorithm works in $(\log n)$ phases. Initially the solution consists of the empty set. During each phase of the algorithm we execute a subroutine Ω to choose a subgraph to add to the solution. The subgraph chosen in each iteration is required to possess two desirable properties. First, it must not increase the budget value of the solution by more than D; second, the solution cost with respect to $\mathbf{B}$ must be no more than OPT_c, where OPT_c denotes the minimum c-cost of a $\mathcal{D}$ bounded subgraph in $\mathbf{S}$. Since the number of iterations of the algorithm is $O(\log n)$ we get a $(\log n, \log n)$-approximation algorithm. The basic technique is fairly straightforward. The non-trivial part is to devise the right subroutine Ω to be executed in each phase. Ω must be chosen so as to be able to prove the required performance guarantee of the solution. We use the solution based decomposition technique [KR93, Ra94, RM+93] in the analysis of our algorithm. The basic idea (behind the solution based decomposition technique) is to use the existence of an optimal solution to prove that the subroutine Ω finds the desired subgraph in each phase.

We now present the specifics of ALGORITHM DCST. The algorithm maintains a set of connected subgraphs or *clusters* each with its own distinguished vertex or *center*. Initially each terminal is in a cluster by itself. In each phase, clusters are merged in pairs by adding paths between their centers. Since the number of clusters comes down by a factor of 2 each phase, the algorithm terminates in $\lceil \log_2 |K| \rceil$ phases with one cluster. It outputs a spanning tree of the final cluster as the solution.

ALGORITHM DIAMETER-CONSTRAINED-STEINER-TREE (DCST):

- *Input:* $G = (V, E)$ - graph with two edge cost functions, c and d, D - a bound on the diameter under the d-cost, $K \subseteq V$ - set of terminals, ϵ - an accuracy parameter.
- 1. Initialize the set of clusters $\mathcal{C}_1$ to contain $|K|$ singleton sets, one for each terminal in K. For each cluster in $\mathcal{C}$, define the single node in the cluster to be the center for the cluster. Initialize the phase count $i := 1$.
  2. Repeat until there remains a single cluster in $\mathcal{C}_i$
     (a) Let the set of clusters $\mathcal{C}_i = \{C_1 \ldots, C_{k_i}\}$ at the beginning of the i'th phase (observe that $k_1 = |K|$).
     (b) Construct a complete graph G_i as follows: The node set V_i of G_i is $\{v : v \text{ is the center of a cluster in } \mathcal{C}\}$. Let path P_{xy} be a $(1+\epsilon)$-approximation to the minimum c-cost diameter D-bounded path between centers v_x and v_y in G. Between every pair of nodes v_x and v_y in V_i, include an edge (v_x, v_y) in G_i of weight equal to the c-cost of P_{xy}.
     (c) Find a minimum-weight matching of largest cardinality in G_i.
     (d) For each edge $e = (v_x, v_y)$ in the matching, merge clusters C_x and C_y, for which v_x and v_y were centers respectively, by adding path P_{xy} to form a new cluster C_{xy}. The node (edge) set of the cluster C_{xy} is defined to be the union of the node (edge) sets of C_x, C_y and the nodes (edges) in P_{xy}. One of v_x and v_y is (arbitrarily) chosen to be the center v_{xy} of cluster C_{xy}. C_{xy} is added to the cluster set $\mathcal{C}_{i+1}$ for the next phase.
     (e) $i := i + 1$.
  3. Let C', with center v' be the single cluster left after Step 2. Output a shortest path tree of C' rooted at v' using the d-cost.
- *Output:* A Steiner tree connecting the set of terminals in K with diameter at most $2\lceil \log_2 n \rceil D$ under the d-cost and of total c-cost at most $(1+\epsilon)\lceil \log_2 n \rceil$ times that of the minimum c-cost diameter D-bounded Steiner tree.

We make a few points about ALGORITHM DCST:

1. The clusters formed in Step 2d need not be disjoint.
2. All steps, except Step 2b, in algorithm DCST can be easily seen to have running times independent of the weights. We employ Hassin's strongly polynomial **FPAS** for Step 2b [Ha92]. Hassin's approximation algorithm for the D-bounded minimum c-cost path runs in time $O(|E|(\frac{n^2}{\epsilon} \log \frac{n}{\epsilon}))$. Thus ALGORITHM DCST is a strongly polynomial time algorithm.
3. Instead of finding an exact minimum cost matching in Step 2c, we could find an approximate minimum cost matching [GW92]. This would reduce the running time of the algorithm at the cost of adding a factor of 2 to the performance guarantee.

We now state some observations that lead to a proof of the performance guarantee of ALGORITHM DCST. Assume, in what follows, that G contains a diameter D-bounded Steiner tree.

Claim 12.9 *Algorithm DCST terminates in* $\lceil \log_2 |K| \rceil$ *phases.*

Proof: Let k_i denote the number of clusters in phase i. Note that $k_{i+1} = \lceil \frac{k_i}{2} \rceil$ since we pair up the clusters (using a matching in Step 2d). Hence we are left with one cluster after phase $\lceil \log_2 |K| \rceil$ and algorithm DCST terminates. ∎

Claim 12.10 *Let* $C \in \mathcal{C}_i$ *be any cluster in phase* i *of algorithm DCST. Let* v *be the center of* C*. Then any node* u *in* C *is reachable from* v *by a diameter-*iD *path in* C *under the* d*-cost.*

Proof: Note that the existence of a diameter D-bounded Steiner tree implies that all paths added in Step 2d have diameter at most D under d-cost. The proof now follows in straightforward fashion by induction on i. ∎

Lemma 12.11 *Algorithm DCST outputs a Steiner tree with diameter at most* $2\lceil \log_2 |K| \rceil \cdot D$ *under the* d*-cost.*

Proof: The proof follows from Claims 12.9 and 12.10. ∎

This completes the proof of performance guarantee with respect to the d-cost. We now proceed to prove the performance guarantee with respect to the c-costs. We first recall the following pairing lemma.

Claim 12.12 *[KR93, RM+93] Let* T *be an edge-weighted 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.*

Proof: A pairing of the marked nodes that minimizes the sum of the lengths of the tree-paths between the nodes paired up can be seen to obey the property in the claim above. ∎

Claim 12.13 *Let* OPT *be any minimum* c*-cost diameter-*D *bounded Steiner tree and let* OPT_c *denote its* c*-cost. The weight of the largest cardinality minimum-weight matching found in Step 2d in each phase* i *of algorithm DCST is at most* $(1+\epsilon) \cdot OPT_c$*.*

Proof: Consider the phase i of algorithm DCST. Note that since the centers at stage i are a subset of the nodes in the first iteration, the centers v_i are terminal nodes. Thus they belong to OPT. Mark those vertices in OPT that correspond to the matched vertices, $v_1, v_2, \ldots, v_{2\lfloor \frac{k_i}{2} \rfloor}$, of G_i in Step 2c. Then by Claim 12.12 there exists a pairing of the marked vertices, say $(v_1, v_2), \ldots, (v_{2\lfloor \frac{k_i}{2} \rfloor - 1}, v_{2\lfloor \frac{k_i}{2} \rfloor})$, and a set of edge-disjoint paths in OPT between these pairs. Since these paths are edge-disjoint their total c-cost is at most OPT_c. Further these paths have diameter at most D under the d-cost. Hence the sum of the weights of the edges $(v_1, v_2), \ldots, (v_{2\lfloor \frac{k_i}{2} \rfloor - 1}, v_{2\lfloor \frac{k_i}{2} \rfloor})$ in G_i , which forms a perfect matching on the set of matched vertices, is at most $(1+\epsilon)\cdot OPT_c$. But in Step 2c of ALGORITHM DCST, a minimum weight perfect matching in the graph G_i was found. Hence the weight of the matching found in Step 2d in phase i of ALGORITHM DCST is at most $(1+\epsilon)\cdot OPT_c$.

∎

Lemma 12.14 *Let OPT be any minimum c-cost diameter-D bounded Steiner tree and let OPT_c denote itsc-cost.* ALGORITHM DCST *outputs a Steiner tree with total c-cost at most* $(1+\epsilon)\lceil\log_2|K|\rceil\cdot OPT_c$.

Proof: From Claim 12.13 we have that the c-cost of the set of paths added in Step 2d of any phase is at most $(1+\epsilon)\cdot OPT_c$. By Claim 12.9 there are a total of $\lceil\log_2|K|\rceil$ phases and hence the Steiner tree output by ALGORITHM DCST has total c-cost at most $(1+\epsilon)\lceil\log_2|K|\rceil\cdot C_D$.

∎

From Lemmas 12.11 and 12.14 we have the following theorem.

Theorem 12.15 *There is a strongly polynomial-time algorithm that, given an undirected graph $G=(V,E)$, with two cost functions c and d defined on the set of edges, diameter bound D, terminal set $K\subseteq V$ and a fixed $\epsilon>0$, constructs a Steiner tree of G of diameter at most $2\lceil\log_2|K|\rceil D$ under the d-costs and of total c-cost at most $(1+\epsilon)\lceil\log_2|K|\rceil$ times that of the minimum-c-cost of any Steiner tree with diameter at most D under d.*

12.6 Exercises

1. Given an undirected graph with nonnegative costs on the edges, the routing cost of any of its spanning trees is the sum over all pairs of nodes, of the cost of the path between the pair in the tree. Give a 2-approximation algorithm for finding a spanning tree of minimum routing cost. (Hint: Consider all the different shortest-path trees).
2. Consider the bicriteria problem (Total edge cost, Routing cost, spanning tree) on an undirected graph where both the total cost and routing costs are computed using the same cost function c_e on the edges. Give a $(O(1),O(1))$ approximation for the problem. (Hint: Reduce the routing cost problem to a shortest path problem, and look in the literature for an appropriate bicriteria result).
3. The buy-at-bulk network design problem [SC+97] is one where the capacities on the edges must be bought in pre-specified bundles for a fixed cost per unit length. In the single-sink single-cable-type version of the problem, we are given an undirected complete graph with a length metric on the nodes, and a specification of the (dollar) cost of buying one unit length of cable with a bandwidth (capacity) of one unit. Moreover, we are given demand amounts at the non-sink nodes that must be routed to the sink node. The problem is to lay down appropriate number of copies of the cable on the edges that supports a routing of all the demands to the sink at minimum total cable cost.

   Two natural lower bounds for the problem arise from *connectivity*, i.e., having to connect all the positive demand nodes to the sink in a Steiner tree with at least one copy of the cable on each edge, and from *routing*, i.e., having to route the demands, each at least along a shortest path to the sink fractionally. Identify the appropriate bicriteria problem that is relevant to the design of an approximation algorithm for the above buy-at-bulk problem; Search the literature for a method that results in an $O(1)$-approximate solution. (Hint: The result of the literature search in the previous problem may be useful for this one as well.)

4. Consider the general strategy for obtaining $(O(\log n), O(\log n))$ approximation algorithms for bicriteria spanning trees on an n-node graph. Suppose the two objectives are (weighted degree, total cost), and these are computed under two distinct cost function (The weighted degree of a node in an edge-weighted spanning tree is the sum of the weights of the edges incident on the node in the tree). What is the subroutine that you need at each stage to get the $(O(\log n), O(\log n))$ result? Do a literature survey to find such a result. What if the objectives are (unweighted degree, total cost), i.e., the case when all edges in the graph have unit weight?

Bibliography

[AB+90] B. Awerbuch, A. Baratz, and D. Peleg, "Cost-sensitive analysis of communication protocols," *Proceedings of the 9th Symposium on Principles of Distributed Computing (PODC)*, pp. 177-187 (1990).

[AK+95] A. Agrawal, P. Klein and R. Ravi, "When trees collide: an approximation algorithm for the generalized Steiner problem on networks," *SIAM Journal on Computing*, vol.24, pp. 440-456 (1995).

[CG82] P. M. Camerini, and G. Galbiati, "The bounded path problem," *SIAM Journal on Algebraic and Discrete Methods* vol. 3, no. 4, pp. 474-484 (1982).

[Ch77] R. Chandrasekaran, "Minimum Ratio Spanning Trees," *Networks,* vol. 7, pp. 335-342, (1977).

[Ch91] C.-H. Chow, "On multicast path finding algorithms," *Proceedings of IEEE INFOCOM 1991*, pp. 1274-1283 (1991).

[CK95] P. Crescenzi and V. Kann, "A compendium of **NP** optimization problems," Manuscript, (1995).

[Fe95] U. Feige, "A threshold of $\ln n$ for approximating set cover," To appear in the *Proceedings of the 28th Annual ACM Symposium on the Theory of Computation* (1996).

[FW+85] A. Frank, L. Wittie, and A. Bernstein, "Multicast communication in network computers," *IEEE Software*, vol. 2, no. 3, pp. 49-61 (1985).

[GG+95] J. L. Ganley, M. J. Golin and J. S. Salowe, "The multi-weighted spanning tree problem," *Proceedings of the First Conference on Combinatorics and Computing (COCOON)*, Springer Verlag, LNCS pp. 141-150 (1995).

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco (1979).

[GW92] M. X. Goemans and D. P. Williamson, "A general approximation technique for constrained forest problems," *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 307-316 (1992). To appear in *SIAM Journal on Computing.*

[Ha92] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36-42 (1992).

[HL+89] J. Ho, D.T. Lee, C.H. Chang and C.K. Wong, "Bounded diameter spanning tree and related problems," *Proceedings of the Annual ACM Symposium on Computational Geometry*, pp. 276-282 (1989).

[Ho95] D. Hochbaum, *Approximation algorithms for NP-hard problems,* D.S. Hochbaum Ed., PWS Publishing Company, Boston, MA (1995).

[Kr56] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the AMS*, Vol. 7, No. 1, pp. 48-50 (1956).

[KJ83] B. Kadaba and J. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Transactions on Communications*, Vol. COM-31, pp. 343-351 (March 1983).

[KR+93] S. Khuller, B. Raghavachari, and N. Young, "Balancing minimum spanning and shortest path trees," *Algorithmica,* vol. 14 (4), pp. 305-321, (1995).

[KP+92A] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Multicasting for multimedia applications," *Proceedings of IEEE INFOCOM 1992* (May 1992).

[KP+92B] 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).

[KP+93] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Multicast routing for multimedia communication," *IEEE/ACM Transactions on Networking*, pp. 286-292 (1993).

[KR93] P. N. Klein, and R. Ravi, "A nearly best-possible approximation for node-weighted Steiner trees," *Journal of Algorithms,* No. 19, pp. 104-115, (1995).

[Me83] N. Megiddo, "Applying parallel computation algorithms in the design of serial algorithms," *Journal of the ACM (JACM)*, vol. 30, pp. 852-865, (1983).

[Ph+93] C. Philips, "The Network Inhibition Problem," *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pp. 776-785, (1993).

[Pr57] R. C. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, pp. 1389-1401 (1957).

[Ra94] R. Ravi, "Rapid rumor ramification: approximating the minimum broadcast time," *Proceedings of the 35th Annual IEEE Foundations of Computer Science*, pp. 202-213 (1994).

[RG95] R. Ravi and M. Goemans, "The constrained spanning tree problem," to appear in the *Proceedings of the 5th Scandinavian Workshop on Algorithmic Theory*, 1996.

[RM+93] 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*, pp. 438-447 (1993). (Expanded version appears as Brown University Technical Report TR-CS-92-58.)

[RS+94] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, "Spanning trees short or small," in *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 546-555 (1994). Journal version to appear in *SIAM Journal on Discrete Mathematics.*

[SC+97] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian, "Buy-at-Bulk Network Design: Approximating the single-sink edge installation problem," *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA '97)*, 619-628 (1997).

[Wa87] A. Warburton, "Approximation of Pareto optima in multiple-objective, shortest path problems," *Operations Research*, vol. 35, pp. 70-79 (1987).

[ZP+94] Q. Zhu, M. Parsa, and W.W.M. Dai, "An iterative approach for delay-bounded minimum Steiner tree construction," *Technical Report UCSC-CRL-94-39*, UC Santa Cruz (1994).