Matching Based Augmentations for Approximating Connectivity Problems*

R. Ravi

Tepper School of Business, Carnegie Mellon University,
Pittsburgh, PA 15213
ravi@cmu.edu

Abstract. We describe a very simple idea for designing approximation algorithms for connectivity problems: For a spanning tree problem, the idea is to start with the empty set of edges, and add matching paths between pairs of components in the current graph that have desirable properties in terms of the objective function of the spanning tree problem being solved. Such matching augment the solution by reducing the number of connected components to roughly half their original number, resulting in a logarithmic number of such matching iterations. A logarithmic performance ratio results for the problem by appropriately bounding the contribution of each matching to the objective function by that of an optimal solution.

In this survey, we trace the initial application of these ideas to traveling salesperson problems through a simple tree pairing observation down to more sophisticated applications for buy-at-bulk type network design problems.

1 Introduction

Approximation algorithms have been traditionally designed and taught on a problem-by-problem basis; Surveys (e.g., [6]) and recent courses and books (e.g., [13]) have approached the area in this way by mainly classifying key results based on a problem-specific basis. As the field matures to provide a rich variety of results, commonalities can be identified to highlight key techniques that become repeatedly useful.

In this survey, we point to one such extremely simple technique that we term MBA, an acronym for Matching Based Augmentation. The two salient features that determine the applicability of the method are that the problem at hand must be a connectivity problem where one tries to connect up various demands (either among themselves or to a common root) in a network, and that the optimal solution can be used to identify an appropriate polynomial-time solvable augmenting subproblem that is a variant of matching. Since the method proceeds by finding such matching iteratively and adding them to augment the solution, the approximation ratio is typically bounded by the number of iterations of the process; Furthermore, since the cost paid by the augmentation

* Supported in part by NSF grant CCF-0430751 and ITR grant CCR-0122581 (The ALADDIN project).

J.R. Correa, A. Hevia, and M. Kiwi (Eds.): LATIN 2006, LNCS 3887, pp. 13–24, 2006.

in each iteration is bounded with respect to the optimal, the method of proof of the performance ratio is primal-based, without relying on any new (lower) bounds on the (minimization) problem to argue the guarantee. Finally, since each augmentation proceeds by matching up components of the current solution, the number of iterations before there is one single component and hence a feasible solution, is logarithmic in the number of demand points that need to be connected. This explains why most approximation algorithms based on this method have logarithmic performance ratio.

We have structured this survey chronologically by describing the applications of the method in the order of their first (typically conference or technical report) publication. In this order, the classic paper of Christofides [2] is the first paper of the sequence to contain most features of the MBA idea: the missing idea is the iterative augmentation. The ATSP approximation of Frieze et al. [3] uses the MBA idea in it complete form to obtain a logarithmic approximation for metric ATSPs. We review these two results in the next section. In the following section, we trace our own work in a series of papers [7, 10, 12, 8, 11] that use this idea in various contexts for NP-hard undirected spanning tree problems. In the next section, we review some more sophisticated uses of the method to solve generalizations of basic connectivity problems so as to route flow under concave cost functions [9, 1, 5]. We close by summarizing the method.

2 The Early Applications

The roots of the matching based augmentation method can be traced back to Christofides' $\frac{3}{2}$-approximation algorithm for the traveling salesperson problem on undirected graphs with metric costs. Recall that in this problem, we are given an undirected (without loss of generality, complete) graph with nonnegative costs obeying the triangle inequality on the edges, and the goal is to find a TSP tour (Hamiltonian cycle that visits every vertex exactly once) of minimum total edge cost.

2.1 Christofides' Algorithm for Metric TSP

Christofides' heuristic [2] first computes a spanning tree T on the graph G. Next, we observe by a simple parity argument on the sum of all degrees in any graph that the number of odd-degree nodes is even. Applying this to the tree T, we see that the number of nodes of odd degree in T is even. We now consider the induced (complete) subgraph on only the odd-degree nodes of T and compute a perfect matching M on this (even-sized) set. Now $T \cup M$ is a connected graph of even degree, which implies that it is Eulerian. An Euler tour of this graph can be shortcut to yield a TSP solution of no higher value (using the triangle inequality property of the metric costs).

While it is clear that the MST T has cost at most that of an optimal tour, bounding the cost of M with respect to an optimal TSP tour requires a little work. Consider an optimal tour and induce it on the odd-degree nodes of T

(short-cutting over the even degree nodes of T). This tour, by the triangle inequality, has cost no more than that of the optimal solution. This induced tour is on an even sized set (by the earlier observation) and hence can be exactly decomposed into two disjoint matchings. The cheaper of these two matchings has cost at most half that of an optimal solution. This in turn upper bounds the cost of the minimum-cost matching M we found on the odd-degree nodes. Overall, the $\frac{3}{2}$ performance ratio is proved.

The key step in the algorithm is to augment the initial tree T by a matching M which can be appropriately bounded by a fraction of the cost of the optimal tour solution. In this way, this algorithm lays out the idea of augmenting a current solution with a matching the cost of which can be bounded by comparing it with an optimal solution. As we shall see, this is the underlying idea of the MBA method.

2.2 The FGM Algorithm for Metric ATSPs

Next, we consider an algorithm due to Frieze, Galbiati and Maffioli [3], henceforth referred to as the FGM algorithm for the asymmetric version of the TSP problem. In this version, a complete directed graph is given with arc costs that are not necessarily symmetric but obey the triangle inequality, and the goal is to find a traveling salesperson directed tour (that visits each vertex exactly once) of minimum total arc cost.

The FGM algorithm is a "greedy" augmentation algorithm that adds arcs to the solution in iterations. It starts with an empty graph in which each node is a singleton component. In each iteration, it adds a collection of cycles that merge these components into larger components. In particular, in the first iteration, it computes a minimum cost directed cycle cover of the nodes and adds it to the solution. This merges the nodes into cycles, and for each cycle a representative node is chosen. In the next iteration, only the induced complete digraph on the representative nodes is considered and a minimum cost cycle cover on the representative nodes is computed and added to the solution. This merges the set of representative nodes (and hence their respective components) in a cycle into a larger component. Notice that every component is strongly connected and Eulerian (every node has indegree equal to outdegree). This proceeds in every iteration by first identifying a representative node in each Eulerian component and computing a minimum cost cycle cover on these representatives to merge components into larger Eulerian components. Finally, when all nodes are in one Eulerian component, we can shortcut an Eulerian tour on all the edges into a Hamiltonian tour of no higher cost using the triangle inequality on the costs.

Two simple observations prove the performance guarantee of $\log_2 n$ for the FGM algorithm on a graph with n nodes: (i) In each iteration the Eulerian components at least halve in number; This is a simple consequence of the fact that every cycle in a cycle cover has length at least two leading to every Eulerian component merging with at least one other such component. (ii) The cost of the cycle cover added in any iteration is at most that of a minimum TSP tour; This follows as a simple consequence of the fact that the minimum TSP tour induced

on the representative nodes in any iteration (and shortcut over the other nodes) is a feasible solution to the cycle cover problem for that iteration, and hence the minimum cover computed has cost no more than that of this optimal TSP solution. Putting these two observations together, we see that the approximation ratio of the FGM algorithm is bounded by the number of iterations, which in turn is at most $\log_2 n$.

The FGM algorithm has all the salient features of the MBA idea: (i) Construct the solution by iterative augmentation using a matching based routine in each iteration (Note that a cycle cover problem on a digraph $G = (V, A)$ is solved by an assignment problem on an auxiliary bipartite graph with node bipartition (V_1, V_2), each of the parts being a copy of V, and edges u_1, v_2 for every arc u, v in A). (ii) The cost of the augmenting solution in each iteration is bounded by that of the optimal by identifying the appropriate matching subproblem to solve the augmentation problem. The overall performance ratio is then proportional to the number of iterations.

3 A Tree Pairing Lemma and Its Applications

In our own work, the MBA method took shape in an unintended context, namely in deriving an approximation algorithm for the node-weighted Steiner tree problem. The conference version of our work [7] proved the performance ratio of the greedy algorithm therein via a simple pairing argument on an even number of nodes in a tree. We recall that here.

Lemma 1. *Let T be a tree and M be an even subset of the nodes of T. There exists a pairing (loosely a "matching") of the nodes of M such that the paths between the pairs in T are edge-disjoint.*

Proof: For a pair (u, v) define the length of the pair to be the number of edges (hops) in T between u and v, The pairing that minimizes the total length has the claimed property. Suppose for a contradiction, two pairs in such a pairing, say (u_1, v_1) and (u_2, v_2) have a common edge e in their paths in T: Breaking up the pairing and re-pairing them using only the paths until e results in a new pairing that reduces the total length of the resulting pairing, contradicting our choice of the pairing.

While being immaterial to our subsequent application of the above lemma, the above proof suggests a constructive method for finding such a pairing: Start with any pairing and repeatedly pick any two pairs that overlap and re-pair them until there are no more such pairs. Since the total length of the pairing reduces at each re-pairing, it is not hard to argue polynomial time termination. Other alternate algorithmic approaches that work include using a minimum length perfect matching procedure on the marked nodes.

3.1 A logarithmic Approximation for MST

We can now use the above lemma to design a simple (but somewhat ridiculous) algorithm for approximating the cost of a minimum spanning tree in an

undirected graph. While there are simple (Kruskal' and Prim's) and even linear time exact algorithms for this problem, the approximation algorithm illustrates some general principles that will explain our subsequent algorithms.

The idea for the approximation is to build the spanning tree in iterations starting with the empty set of edges; The aim is to reduce the number of connected components at the end of each iteration to a constant fraction (typically half) of the number at the beginning of the iteration. For spanning trees, the simplest way to accomplish this is to ensure that every component connects with at least one other component via the edges added in a typical iteration.

How can one arrive at the polynomial time subproblem that accomplishes the component reduction but whose solution can be bounded against an optimal solution? This is the crux of applying the MBA method and the answer depends on the problem at hand.

Let's develop some common notation that will be useful for the rest of this section. Let the total number of iterations for the MBA based algorithm be denoted by τ (typically, $\tau = O(\log n)$). In iteration $t \in \{1, 2, \ldots, \tau\}$, let the set of edges added to augment the solution be denoted E_t, and let the set of connected components at the end of this iteration be denoted $\mathcal{C}_t$ with the connected components labeled $C_t(1), C_t(2), \ldots, C_t(k_t)$, where k_t is the number of connected components in $\mathcal{C}_t$. For example, $\mathcal{C}_0 = V$ with $k_0 = |V| = n$, while $\mathcal{C}_\tau$ is one single connected component with $k_\tau = 1$.

To return to the question about the subroutine to employ at each iteration, we reason as follows: Consider an optimal MST, T^* say, and at the start of iteration $t+1$, we look at the components of $\mathcal{C}_t$ and contract them to supernodes in T^*. The edges of T^* now form a potentially cyclic set of edges with some self loops and multiedges on the node set $\mathcal{C}_t$. We can remove cycles (and self-loops) to finally get a tree (call it $T^*(t)$) on this set of supernodes that use only edges of T^* and hence of total cost no more than the optimum. Now we can apply the tree pairing lemma to $T^*(t)$ (Assume for now that the number of supernodes in $T^*(t)$ is even for otherwise we can omit an arbitrary supernode). The tree-pairing lemma shows how the supernodes can be paired off using edges of T^* and be connected between these pairs. The resulting matching problem that can be used to solve the resulting connection problem is to connect each component of $\mathcal{C}_t$ with another at minimum total cost of all such pairwise connections. Note that even though the original costs may not be metric, we can use a metric completion between supernodes in solving this matching problem: Indeed, if an edge used in the matching is not a direct edge but one in the metric completion, we can use the path of this cost to connect the two endpoints, satisfying the connectivity feasibility requirement of this iteration.

To summarize, in iteration $t+1$, we compute the metric completion of the supernodes in $T^*(t)$ and solve a minimum cost perfect matching problem (assuming the number of supernodes in it is even). For every edge in the matching, we add the path in the graph of this cost during this iteration. The following two lemmas are now immediate.

Lemma 2. *The number of iterations of the MBA-based algorithm is* $O(\log n)$.

The proof follows from the observation that all but one component are paired off in every iteration this reducing the number of components in any iteration by at least a fraction of $\frac{2}{3}$. Starting with $|V| = n$ components, the number of iterations is bounded as above.

Lemma 3. *The cost of edges added at every iteration at most that of an optimal solution T^*.*

The proof of this lemma uses the metric completion on the components of $\mathcal{C}_t$ and using the induced solution $T^*(t)$ and the tree-pairing lemma on it, identifies a matching of cost at most $T^*(t)$ that pairs up the components. Since a minimum cost perfect matching subroutine finds such a pairing of minimum cost, its cost is no more that that of $T^*(t)$ as stated.

Putting the above two lemmas together and observing that at the end of the last iteration, we have added a set of edges that form a single connected component, we can delete edges as required to get a final spanning tree of cost no more than the number of iterations times that of T^*. Along with the observation that the subproblem we set up at each iteration is polynomial-time solvable we have the following theorem.

Theorem 1. *The MBA-based algorithm using a minimum cost perfect matching subroutine at each iteration outputs a spanning tree of total cost $O(\log n)$ times the minimum.*

Since the tree pairing lemma works only on a subset of nodes, the results in the following sections all apply to finding Steiner trees that connect a subset of the nodes (called terminals) rather than the whole node set as in a spanning tree. We restrict our discussion to spanning trees for the sake of simplicity and reduced notation, but note that the $O(\log n)$ factors in the treatment below is typically reduced to $O(\log k)$ where k is the number of terminals in the Steiner tree problem.

3.2 Degree Bounded MSTs

The first problem using the MBA framework is the degree-constrained minimum spanning tree problem: Given integer degree budgets $B_v > 0$ for every vertex v of an undirected graph with nonnegative edge costs, the goal is to find a spanning tree of minimum total cost obeying all the degree bounds (if it exists), i.e., the degree of node v in the tree is at most B_v. This problem generalizes minimum-cost TSP paths by setting the budget to one at the endpoints and two elsewhere. Furer and Raghavachari [4] used a matching based approach to derive the first approximation algorithm for a special case of the problem with all edge costs being either one or infinity (the unweighted graph case), and the solution output by their method used a degree-constrained subgraph subroutine to get an $O(\log n)$ approximation ratio for all the degree budgets simultaneously (i.e., if B_v is feasible for all v, their solution has degree $O(\log n \cdot B_v)$ at v for all v. Their algorithm can be seen as an early application of the MBA method.

The degree constrained MST problem was first addressed in our work [10] where the tree-pairing lemma was used to identify a matching subproblem to connect up components in each iteration [1].

The algorithm in [10] follows the same outline as that for MSTs in the previous subsection. The subroutine at each iteration must be tailored to add a subgraph that induces degree no more than about B_v at any node v, and has cost no more than that of an optimum solution, while merging components in pairs. The resulting matching problem turns out to be a bit more sophisticated than that for MST as expected since it handles two different objectives, namely node degrees and edge costs. The subroutine builds a bipartite graph with the original nodes on the left part and the current connected components $\mathcal{C}_t$ on the right part. Original graph edges are duplicated to go between each vertex endpoint on the left part to the component on the right part containing the other endpoint. The subroutine is now to choose a minimum cost set of edges that have at least one edge leaving every component (on the right part) but have degree at most say $2B_v$ at any node v on the left part. The tree pairing lemma guarantees that the paths between the pairings induce degree at most twice the original degree. While the counterpart of Lemma 2 is immediate, we have the following version of Lemma 3.

Lemma 4. *The cost of edges added at every iteration at most that of an optimal solution T^*, while the degree added to any node v in any iteration is at most $2B_v$.*

Finally we get the following theorem.

Theorem 2. *[10] The MBA-based algorithm using a minimum-cost degree-constrained subgraph subroutine at each iteration outputs a spanning tree of total cost $O(\log n)$ times that of a minimum cost tree obeying the degree bounds; Moreover, the spanning tree output has degree at most $O(\log n \cdot B_v)$ at node v for all vertices v.*

3.3 Diameter Bounded MSTs

Next, we turn to a "cost-diameter" version of the MST problem: Given a nonnegative length l_e and a nonnegative cost c_e for every edge e of an undirected complete graph, the goal is to find a cheap tree (in terms of total cost) and also low diameter (in terms of lengths). In a particular budgeted version of the problem, we are given a bound L on the total (length) diameter of the spanning tree to be output and the goal is to find such a spanning tree of minimum total edge cost. This minimum cost-diameter spanning tree problem can be easily shown to be NP-hard [8], as is a cost-radius version of the problem. In the cost-radius version, we are given a root node r, and a bound R on the total length of any path in the output tree from r to any node (hence the name radius, in terms of

[1] While this treatment has been completely worked out in the conference version of our paper [10], the journal version [12] uses a different greedy approach that can also handle node weights in a generalized version of the basic problem.

the length function). We will relate this cost-radius spanning tree problem to a cost-distance version in the next section.

Let us apply the MBA based method to find a minimum cost-diameter spanning tree. At each iteration, we must merge components but with two different goals in mind: the total cost of the matching paths added in the iteration must be at most that of an optimal solution, and the diameter of every path added in the matching should also not exceed the bound L. A further complication is introduced in keeping the total diameter of the final solution bounded with respect to L. For this reason, we simple promote one of the two endpoints of the matched pairs as a representative for its connected component in the next iteration to control the growing radius of the component. Applying the tree pairing lemma to the set of representatives in an optimal tree (of diameter L and total cost C^* say), we can pair the representatives using paths of length at most L each and of total cost at most C^*.

This leads to the following matching subroutine in each iteration. We have a set of connected components, each with a representative. We build an auxiliary graph only on the representatives connecting every pair of representatives by an edge that represents paths of length at most L. Furthermore, we want the cost of these paths to be minimum under the length constraint. For this, we solve a constrained shortest-path problem between this pair of representatives: in particular, we find the minimum cost of a path of total length at most L between these representatives. This problem is itself weakly NP-hard but a scaled adaptation of Djikstra's algorithm gives a PTAS for this path cost computation (i.e, we can get a $(1+\epsilon)$-approximation to the minimum cost path of length at most L in polynomial time for any fixed $\epsilon > 0$). After filling in all these path costs between representatives, we find a minimum cost perfect matching under these costs. Note that this pairs up components via their representatives using paths of length no more than L^* and nearly minimum total cost.

As in Lemma 2, the guarantee on the number of iterations follows from the pairing property of the paths added in every iteration. We also have the following guarantee on the cost and diameter of components at the end of every iteration.

Lemma 5. *The cost of edges added at every iteration at most* $(1+\epsilon)$ *times that of an optimal solution* T^* *for some fixed* $\epsilon > 0$*, while the diameter of any connected component at the end of iteration* t *under the length function is at most* $2tL$*.*

The bound on the diameter follows from an inductive argument while the cost guarantee is a consequence of the tree-pairing lemma. To obtain the final solution, we observe that even though the set of edges we may have added may form cycles, we can choose a minimum radius tree (under the length function) rooted at the representative of the final component. This tree obeys the bounds in the next theorem.

Theorem 3. *[8] The MBA-based algorithm using a minimum-cost length constrained subgraph subroutine at each iteration outputs a spanning tree of total cost* $O(\log n)$ *times that of a minimum cost tree obeying the diameter bounds; Moreover, the spanning tree output has diameter at most* $O(\log n \cdot L)$*.*

3.4 Degree and Diameter Bounded Trees

A third application of the tree-pairing lemma to formulate an MBA based approximation came in the unlikely guise of minimizing the broadcast time in an undirected graph. In this problem, we are given a undirected graph with a root node r containing a message to be broadcast to all the nodes in the graph. At each time step, every node that has a copy of the message can transmit it to one of the (uninformed) neighbors, in the so-called telephone model. The goal is to find a scheme for broadcasting the message to all nodes in the minimum number of time steps. In the first poly-logarithmic approximation algorithm for this problem [11], we showed how to reduce this problem to one of finding a spanning tree with simultaneously low diameter and low maximum node degree. The *poise* of a spanning tree in an undirected graph captures this notion and is defined as the sum of the diameter and the maximum degree. A spanning tree of an undirected graph on n nodes with poise ρ can be used to broadcast a message from any root node within $O(\frac{\log n}{\log \log n} \cdot \rho)$ time steps.

The problem of finding spanning trees with minimum poise can be attacked using the MBA method. At each iteration, the matching based subroutine is require to add paths between matched components that have low diameter (number of hops) as well as induce low degree on any node in the graph. We can use the idea of promoting representatives from the previous subsection (for minimum cost-diameter spanning trees) to control the diameter of the connected components at each iteration. Applying the tree-pairing lemma to the representatives on an optimal tree, we can infer that there is a matching between them using paths of length at most the optimal poise such that the maximum degree induced by these paths at node is also at most the optimal poise. This motivates a corresponding matching problem of pairing up the representatives using short paths with low congestion at any node.

To set up this problem so as to control for the maximum degree of any node induced by the set of matching paths, we use ideas from minimizing congestion in routing integral multicommodity flow, and formulate a linear programming problem to which we can apply randomized rounding. To summarize, the set of representatives from the connected components at each iteration are the sources of multicommodity flow that sinks at any of the other representatives. Furthermore, the length of any of these flow paths is bounded by a given budget (on the poise). An LP solution to the resulting problem of minimizing the node congestion can be rounded randomly to get a near-optimal integral solution. The tree-pairing lemma again provides a proof that there is an integral (and hence LP) solution for the right guess value of the poise with maximum node congestion also at most this poise. The integral rounded solutions can be used to find appropriate matching paths between components in a way that the diameter only increases linearly with the number of iterations. A slightly more careful choice of pairing paths still guarantees the bounds of Lemma 2 while we can get the following analogue of the cost bounding lemma.

Lemma 6. *If there is a tree of poise ρ in the input graph, the LP rounding method with subsequent careful choice of matching paths induces degree at most*

$O(\rho + \log n)$ at any node in each iteration, while the diameter of any connected component at the end of iteration t under the length function is at most $2t\rho$.

Noting that the minimum poise of any spanning tree of an n-node graph is $\Omega(\frac{\log n}{\log \log n})$, we get the following result.

Theorem 4. *[11] The MBA-based algorithm using randomized rounding of a length-constrained node-congestion minimizing LP at each iteration outputs a spanning tree of poise $O(\frac{\log^2 n}{\log logn})$ times the minimum.*

This subsequently leads to the same performance guarantee for the minimum broadcast time problem as shown in [11].

4 Algorithms Inspired by MBA

In this section, we briefly review two lines of work that have used the MBA technique but pushed it to a whole new level. While the underlying matching is recognized as a vehicle to argue the cost incurred by the algorithm by charging it against an optimal solution, these methods typically employ randomization (in their simplest versions) to show expected guarantees on the cost of one iteration. Logarithmic guarantees follow using the same basic line of argument as for the MBA method.

4.1 Cost-Distance Network Design

The cost-distance network design problem is a variant of the set of distance-constrained minimum-cost spanning tree problems introduced in Section 3.3. In this problem, we are given a nonnegative length l_e and a nonnegative cost c_e for every edge e of an undirected complete graph as well as a root node r. In the simplest version, the goal is to find a spanning tree that minimizes the sum of the costs of the edges in the tree (under the c-function) and the distances in the tree (under the l-function) from the root to all the nodes.

The algorithm given by Myerson et al. [9] for this simple version is to define a composite weight function that is the sum of the cost and length for each edge. The algorithm then finds a near-perfect minimum weight perfect matching (ignoring the root and connecting to it only in the last iteration) and chooses one of the two endpoints to be a representative for the whole component randomly. As in the MBA algorithms, these paths are added and the process continues until a tree is obtained.

As in the MBA method, the proof of performance ratio proceeds by showing that the expected cost of eventually connecting all the vertices to the root via the matching added in one iteration is bounded by a constant factor times that of the optimal solution. The randomization allows one to argue that as the iterations proceed that the subproblems on the representatives (which can be thought of as aggregating the demand of all nodes in its component) has expected cost at

most that of an optimal solution. Using a similar line of proof as in the MBA method, a performance ratio of $O(\log n)$ follows for this algorithm.

A derandomization of the method along with links to the integrality gap of a natural LP relaxation was provided in [1]. This derandomizing procedure proceeds via the method of conditional probabilities using an LP relaxation; The underlying matching problem is solved motivated by an argument that can be viewed as a more sophisticated matching version of the tree pairing lemma arising in the context of the new composite cost function.

4.2 Simultaneous Optimization for Concave Costs

A further generalization was studies by Goel and Estrin in [5]. In this version, we are given an undirected graph with a root r and a nonnegative cost c_e for every edge e. The goal is to find an "aggregation" tree that collects information from all the nodes to the root. The cost of the tree depends on the aggregation functions on the edges. Let f be a real-valued function defined on non-negative real numbers that is concave and nondecreasing. The cost of an edge e is then $c_e f(flow_e)$ where $flow_e$ is the flow routed through e, in this case the total number of nodes in the subtree under e, when the solution is rooted at r.

Goel and Estrin use a variant of the MBA method to prove a surprising result: There is a tree that is simultaneously near-optimal for all concave aggregating functions for a given undirected graph with costs. This tree is none other than a MBA-based tree constructed in iterations based on the cost function c on the edges. Assuming that the number of nodes n is a power of two. This method simply finds a minimum-cost perfect matching on the nodes and chooses one endpoint as a representative with probability half, and continues until all nodes are connected in a spanning tree.

The proof of performance of this aggregating tree for any fixed concave aggregating function proceeds in a similar way as for the cost-distance problem. First, the expected cost of the rerouted instances is bounded by that of the optimal. Second, the expected aggregated routing cost of the matching edges added in each iteration is bounded by the cost of the optimal solution. To prove the result for general functions, the method employed is to carry out the analysis in terms of some basis aggregation functions (also called "atomic" functions in [5]) that aggregate linearly up to some power of two. Any concave aggregating function's cost is written as a scaled contribution from an appropriate basis function, which are then used in a style similar to that for a fixed function to argue the final result. At this level, while the basic algorithm and outline of the proof technique (using an optimal solution to bound expected cost of the current augmentation) are as in the MBA based methods, this application requires a much more involved argument.

5 Summary

We have reviewed various applications of a simple construction heuristic idea with the augmentations coming from a matching-like subroutine that is inspired

by a very simple tree-pairing lemma. Recent refinements replace the tree-pairing with a randomized demand redistribution for reallocation of the cost of the current iteration to that of an optimal solution. The simple idea of using an optimal solution appropriately to derive an augmentation of the solution has been effectively used in a variety of contexts, but we hope the reader is left with a sense of commonality in these applications for network design problems.

References

1. C. Chekuri, S. Khanna, and S, Naor. A deterministic algorithm for the COST-DISTANCE problem. *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 232-233, 2001.
2. N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388, Graduate School of Industrial Administration, CMU, 1976.
3. A.M. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks* 12 (1982) 23-39.
4. M. Fürer and B. Raghavachari. An NC approximation algorithm for the minimum-degree spanning tree problem. *Proceedings of the 28th Annual Allerton Conference on Communication, Control and Computing* 274-281, 1990.
5. A. Goel and D. Estrin. Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk. *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.
6. D. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* P.W.S, 1997.
7. Philip N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *J. Algorithms*, 19(1):104–115, 1995. An early version appeared in the *Proceedings of the Annual MPS conference on Integer Programming and Combinatorial Optimization*, 1992.
8. M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt. Bicriteria network design problems. *J. of Algorithms*, 28, 142171, 1998.
9. A. Meyerson, K. Munagala, and S. Plotkin. Cost-Distance: Two Metric Network Design. *Proceedings of the 41st Annual IEEE Symposium on the Foundations of Computer Science*, 2000.
10. Many birds with one stone: Multi-objective approximation algorithms. R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III. *Proceedings of the ACM Symposium on the Theory of Computing*, 438-447, 1993.
11. R. Ravi. Rapid Rumor Ramification: Approximating the minimum broadcast time. *Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science*, 202–213, 1994.
12. Approximation Algorithms for Degree-Constrained Minimum-Cost Network-Design Problems. R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, H. B. Hunt III. *Algorithmica* 31(1), 58-78, 2001.
13. Vijay V. Vazirani. *Approximation Algorithms.* Springer-Verlag, Berlin, 2001.