Improving Steiner Trees of a Network Under Multiple Constraints
(Extended Abstract)

S.O. Krumke [1] H. Noltemeier [1] M.V. Marathe [2]
R. Ravi [3] S.S. Ravi [4]

January 22, 1997

Abstract

We consider the problem of decreasing the edge weights of a given network so that the modified network has a Steiner tree in which two performance measures are simultaneously optimized. We formulate these problems, referred to as *bicriteria network improvement problems*, by specifying a budget on the total modification cost, a constraint on one of the performance measures and using the other performance measure as a minimization objective. Network improvement problems are known to be NP-hard even when only one performance measure is considered. We present the first polynomial time approximation algorithms for bicriteria network improvement problems. Our approximation algorithms are for two pairs of performance measures, namely (diameter, total cost) and (degree, total cost). These algorithms produce solutions which are within a logarithmic factor of the optimum value of the minimization objective while violating the constraints only by a logarithmic factor. Many of our approximation results can be extended to more general network design problems.

[1] Department of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany. Email: {krumke,noltemei}@informatik.uni-wuerzburg.de.

[2] Los Alamos National Laboratory, P.O. Box 1663, MS M986, Los Alamos, NM 87545, USA. Email: madhav@c3.lanl.gov. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

[3] GSIA, Carnegie Mellon University, Pittsburgh, PA. Email: ravi+@andrew.cmu.edu.

[4] Department of Computer Science, University at Albany – SUNY, Albany, NY 12222, USA. Email: ravi@cs.albany.edu.

1 Introduction

The problem of modifying an edge weighted network so as to improve some measure of its performance arises in a number of applications including high speed communication networks [22], video on demand [25], teleconferencing [23, 24] and VLSI design [10, 11, 32]. In such problems, there is usually a cost associated with reducing the weight of any edge. Further, there is also a budget on the total reduction cost. Some examples of performance measures that can be improved by reducing the edge weights are total cost of a minimum spanning tree, bottleneck cost of a spanning tree and diameter. Other similar performance measures can be formulated by replacing the requirement of a spanning tree by some other connectivity requirement (e.g. Steiner tree, k node or k edge connected subgraph). The goal is to find a reduction strategy so that the total cost of the reduction satisfies the budget constraint and the modified network has the best possible performance measure over all possible reduction strategies that obey the budget constraint. Problems where the performance of the modified network is specified in terms of one measure have been considered in [20].

In this paper, we study the problem of finding optimal improvement strategies when the performance of the modified network is characterized by a pair of parameters rather than one. We refer to such problems as *bicriteria network improvement* problems. Specifically, the problems considered here include finding budget constrained optimal improvement strategies for two pairs of performance measures, namely (diameter, total cost) and (degree, total cost), of a spanning tree in the modified network. Precise formulations of these problems are presented in the next section.

We observe that in general the problem of computing an optimal reduction strategy for modifying the network as above is NP-hard and present the first polynomial time approximation algorithms for the problem. Our techniques also lead to improved approximation algorithms for the class of treewidth-bounded graphs. Many of our results can be extended to obtain approximation algorithms for more general network design problems such as those considered in [16, 17, 1].

The remainder of this paper is organized as follows. Section 2 presents preliminary definitions and problem formulations. In Section 3 we summarize the results obtained in this paper and discuss briefly some related work. Sections 4 and 5 present our approximation algorithms. Finally, Section 6 briefly states hardness results for the problems considered in this paper.

2 Problem Definitions

Let $G = (V, E)$ be an undirected graph. Associated with each edge $e \in E$, there are four nonnegative values as follows: $\ell(e)$ denotes the *length* or the *weight* of the edge e, $c(e)$ denotes the *cost* of reducing the length of edge e by one unit, $\ell_{\min}(e)$ denotes the *minimum length* to which the edge e can be reduced.

A *reduction strategy* (or simply *reduction*) on the edges of G specifies how to reduce the ℓ-lengths of each edge e to a value in the range $[\ell_{\min}(e), \ell(e)]$. Given a budget $\mathcal{B}$, we define a *feasible reduction* to be a nonnegative function r defined on E with the following properties: For all edges $e \in E$, $\ell(e) - r(e) \geq \ell_{\min}(e)$ and $\sum_{e \in E} r(e) \cdot c(e) \leq \mathcal{B}$. If r is a (feasible) reduction in G, we can consider the graph G with the edge weights given by the "reduced lengths", namely, $(\ell - r)(e) := \ell(e) - r(e)$.

Let $K \subseteq V$ be a set of distinguished nodes called *terminals*. Any subgraph T of G which is a tree and which contains all the terminals from K is said to be a *Steiner tree* of G for K. The *total length* of a tree T under the weight function ℓ, denoted by $\ell(T)$, is defined to be the sum of the lengths $\ell(e)$ of the edges $e \in T$. We denote the total weight of a minimum total length Steiner tree with respect to ℓ by $\text{Steiner}_G(\ell)$. We omit the graph G in the subscript whenever such an omission does not cause any ambiguity.

Given an additional edge weight function d which is independent of ℓ, the *diameter* of a tree T under d is the length of a longest simple path in T with respect to d. Let $\mathcal{D}$ be a bound on the d-diameter of a Steiner tree. Then $\text{Steiner}_G(\ell, d, \mathcal{D})$ denotes the total length $\ell(T)$ of a Steiner tree T in G such that T has diameter at most $\mathcal{D}$ under the d-function, and the value $\ell(T)$ is a minimum among all Steiner trees satisfying the above diameter constraint. Similarly, $\text{Steiner}_G(\ell, \deg(v), b)$ denotes the total length $\ell(T)$ of a Steiner tree T in G such that the maximum degree of a vertex in T is at most b. We now state the problems studied in this paper.

Definition 2.1 The *Diameter and Budget Constrained Minimum Total Cost Steiner Tree Problem* (Dia-BCST) is to find a feasible reduction r such that $\text{Steiner}_G(\ell - r, d, \mathcal{D})$ has the least possible value and to output a corresponding Steiner tree.

Definition 2.2 The *Degree and Budget Constrained Minimum Total Cost Steiner Tree Problem* (Deg-BCST) is to find a feasible reduction r such that $\text{Steiner}_G(\ell - r, \deg(v), b)$ has the least possible value and to output a corresponding Steiner tree.

Extensions of the above definitions to other general one-connected subgraph problems studied in [17, 1] are straightforward.

It follows easily from the NP-hardness of the classical Steiner tree Problem and the problem of finding a minimum weight tree subject to a degree constraint (cf. [15]) that both Dia-BCST and Deg-BCSTproblems are NP-hard. Moreover, even if one is given an optimal reduction r^* for one of the problems, it remains hard to find a corresponding optimal tree in the modified network. These hardness results continue to hold, even if we allow a violation of the budget-constraint. We briefly discuss such hardness results in Section 6.

Given the hardness of the problems, it seems appropriate to use a slightly relaxed notion of an approximation algorithm. A *multicriteria approximation algorithm* (cf. [29, 27, 20]) finds a solution which may be suboptimal in general and which may violate the given constraints within a certain range. Naturally, its *performance* is then measured in terms of the deviation from the optimal value and the extent to which the constraints are violated.

Definition 2.3 Let Π be an optimization problem where the goal is to find a solution x^* minimizing the objective function goal_Π subject to constraints $g_i(x^*) \le G_i$, $i = 1, \ldots, k$. A $(k+1)$-*criteria approximation algorithm* for Π with performance $(\alpha_0, \alpha_1, \ldots, \alpha_k)$ is a polynomial time algorithm which, given any instance of Π, finds a solution x such that $g_i(x) \le \alpha_i G_i$ $(i = 1, \ldots, k)$ and $\text{goal}_\Pi(x) \le \alpha_0 \text{OPT}$, where OPT denotes the minimum value of a solution satisfying all the constraints.

In this paper we will be concerned with *tricriteria approximation algorithms*. Consider, e.g. Dia-BCST. The objective function of the problem is the length $(\ell - r)(T)$ of the Steiner tree in the modified network. We consider the diameter constraint to be the first constraint, the budget constraint as the second one. A tricriteria approximation algorithm with performance $(\alpha_0, \alpha_1, \alpha_2)$ is required to find a reduction of cost at most $\alpha_2 \mathcal{B}$ and a tree T of diameter at most $\alpha_1 \mathcal{D}$ whose modified length $(\ell - r)(T)$ is at most α_0 times the optimum.

3 Summary of results and related research

Given the hardness of Dia-BCST and Deg-BCST, we focus on approximation algorithms for these problems.We obtain the *first* polynomial time approximation algorithms for the Dia-BCST and Deg-BCST problems with guaranteed performance. The results obtained in this paper include the following.

1. We give a polynomial time approximation algorithm for the problem Dia-BCST. Given any instance of the Dia-BCSTproblem, the algorithm outputs a reduction r and a Steiner tree T such that:

   (a) The length of T with respect to the modified lengths $\ell - r$ is $\mathcal{O}(\log_2 |K|) \cdot \mathrm{OPT}$.
   (b) The d-diameter of T is $\mathcal{O}(\log_2 |K|) \cdot \mathcal{D}$.
   (c) The cost of the reduction r is at most $\mathcal{O}(\log_2 |K|) \cdot \mathcal{B}$.

   Thus, the algorithm is a $(\mathcal{O}(\log_2 |K|), \mathcal{O}(\log_2 |K|), \mathcal{O}(\log_2 |K|))$ tricriteria approximation algorithm.

2. We give a polynomial time approximation algorithm for the problem Deg-BCSTwith a $(\mathcal{O}(\log_2 \frac{|K|}{b}), \mathcal{O}(\log_2 \frac{|K|}{b}), \mathcal{O}(\log_2 \frac{|K|}{b}))$ performance.

As far as we know, the problems considered in this paper have not been previously studied. Recently, Frederickson and Solis-Oba [13] considered the problem of increasing the weight of the minimum spanning tree in a graph subject to a budget constraint where the cost functions are assumed to be linear in the weight increase. In contrast to the work presented here, this problem turns out to be much easier and is shown to be solvable in strongly polynomial time. Berman [3] considers the problem of shortening edges in a given tree to minimize its shortest path tree weight. This problem is solvable in polynomial time by a straightforward greedy algorithm. Phillips [28] studies the problem of finding an optimal strategy for reducing the capacity of the network so that the residual capacity in the modified network is minimized. The problems studied here and in [28, 3] can be broadly classified as types of multicriteria problems. Recently, there has been substantial work on finding efficient approximation algorithms for a variety of bicriteria problems (see [21, 18, 27, 29, 30, 31, 32] and the references therein).

4 Approximation Algorithm for Dia-BCST-Problem

In this section, we discuss our approximation algorithm for Dia-BCST. We first present an informal description of the algorithm in order to present the main ideas behind the steps. We then proceed with a more detailed presentation and depict our approximation algorithm, called Improve-Dia-BCST, in the usual pseudo code.

Our algorithm works roughly as follows: We first modify the graph by replacing each edge with a set of parallel multiedges[5], which reflect the improvement in the length value for each discrete increase in money spent in a reduction r on the particular edge. In order to establish the performance guarantee, we must show that the modification of the graph preserves the structure of the problem.

The next main step of the algorithm consists of $\mathcal{O}(\log |K|)$ *phases*, where K is the set of terminals. Initially the solution consists of the empty set. During each phase of the algorithm

[5]We will sketch in Section A of the appendix how multiedges can be avoided by replacing each edge of the original graph by a certain subgraph.

we choose a set of edges to add to the solution. The set of edges chosen in each iteration is required to possess three desirable properties: (1) The reduction cost spent in each iteration is no more than $\mathcal{B}$; (2) The diameter value with respect to d must not increase by more than $\mathcal{D}$; (3) Finally, the solution cost with respect to the shortened ℓ-cost must be no more than OPT, where OPT denotes the optimally improved total length of a diameter $\mathcal{D}$-bounded tree under a budget $\mathcal{B}$ on the cost of improvement.

Since the number of iterations of the algorithm is $\mathcal{O}(\log |K|)$, this will lead to an approximation algorithm which violates the budget and the diameter constraint by a factor of at most $\mathcal{O}(\log |K|)$ and which is within a factor of $\mathcal{O}(\log |K|)$ of the optimal solution. We use a *solution based decomposition technique* (cf. [29]) in the analysis of our algorithm. Its basic idea is to use the existence of an optimal solution to prove that in each phase, it is possible to choose a good set of edges. We now refine the specification of our algorithm Improve-Dia-BCST.

Procedure Transform(ε)

1 **for** each edge $e = (u, v)$ in the graph let b_e be chosen so that $(1+\varepsilon)^{b_e} \leq \ell(e) - \ell_{\min}(e) \leq (1+\varepsilon)^{b_e+1}$.

2 Add $b_e + 2$ parallel edges e^k, $k = -1, 0, \ldots, b_e$, between u and v.

3 For $k \geq 0$, the edge e^k has length $\ell'(e^k) = \ell(e) - (1+\varepsilon)^k$, cost $r'(e^k) = (1+\varepsilon)^k c(e)$ and d-weight $d(e^k) := d(e)$, while the edge e^{-1} has length $\ell(e)$, cost 0 and weight $d(e)$.

Figure 1: Procedure to discretize the problem in the first main step and transform G into G'.

Main Step 1: Discretize the problem by using the transformation procedure Transform shown in Figure 1. This transformation builds a graph G' from the original graph G in the following way: Each edge $e = (u, v)$ with length $\ell(e)$ and minimum length $\ell_{\min}(e)$ is replaced by a set of parallel edges. These edges have discrete lengths from the interval $[\ell(e), \ell_{\min}(e)]$. An edge e^t of length $\ell'(e^t) := \ell(e) - t$ (where t is the discrete step size) is assigned the cost of reducing e by t as cost; i.e., the cost $r'(e^t)$ of edge e^t is $t \cdot c(e)$. The d-weights for the edges are all the same and coincide with that of e. To avoid additional notation, we also use d to denote these weights. This way, each edge e' in G' is assigned three numbers $\ell'(e')$, $r'(e')$ and $d(e')$.

Main Step 2: 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 of the second main step, we set up an auxiliary graph H using the constrained shortest path algorithm of [18]. We then solve a variant of a constrained matching problem in H and, finally, merge the clusters in pairs by adding paths between their centers. Since the number of clusters decreases by a factor of 2 due to the merging in each phase, the algorithm terminates in $\lceil \log_2 |K| \rceil$ phases with one cluster.

Main Step 3: We take a tree T' in G' rooted at the center of the single cluster left at the end of the second main step with small diameter under the d-costs.

Main Step 4: In the final step, we use the information from the three values associated with each edge in T' to construct a reduction strategy r and a Steiner tree T for the original graph G. We output r and T.

A complete specification of our Algorithm Improve-Dia-BCST is shown in Figure 2 on the next page.

Before analyzing Algorithm Improve-Dia-BCST, we make a few more comments on it. The clusters formed in Step 5e need not be node (or edge) disjoint. All steps, except Step 5(c)i

Algorithm Improve-Diameter-Constrained-Steiner-Tree (Improve-Dia-BCST):

Input: $G = (V, E)$ - Graph with four edge cost functions, ℓ, $\ell_{\min}$, c and d.
$\mathcal{D}$ - Bound on the diameter under the d-weight,
$\mathcal{B}$ - Bound on the amount of money spent in updating the graph,
$K \subseteq V$ - Set of terminals, and
ε - Accuracy parameter.

1. Call Transform(ε) to obtain a new graph $G' = (V, E')$ with weights $\ell'(e)$, $r'(e)$, $d(e)$ on the edges $e \in E'$.
2. Let $M := \lceil \log_{(1+\varepsilon)} \mathcal{B} \rceil + 1$.
3. For $0 \le j \le M$, define $\mathcal{B}_j := (1+\varepsilon)^j$. Further, define $\mathcal{B}_{-1} := 0$. Notice that the given budget $\mathcal{B}$ satisfies $(1+\varepsilon)\mathcal{B} \in [\mathcal{B}_{-1}, \mathcal{B}_M] = [0, \mathcal{B}_M]$.
4. Initialize the set of clusters $\mathcal{C}_1$ to contain $|K|$ singleton sets, one for each terminal in K. For each cluster in $\mathcal{C}_1$, define the single node in the cluster to be the center for the cluster. Initialize the phase count $i := 1$.
5. Repeat until there is only one cluster in $\mathcal{C}_i$
   (a) Let the set of clusters at the beginning of the i'th phase be $\mathcal{C}_i = \{C_1 \ldots, C_{k_i}\}$ (observe that $k_1 = |K|$).
   (b) Construct a (complete) multigraph G_i with node set V_i defined by $V_i := \{\ v :\ v$ is the center of a cluster in $\mathcal{C}\ \}$ and multiedges as follows:
   (c) For $-1 \le j \le M$ do the following:
      i. Let the path $P_{xy}(\mathcal{B}_j)$ be a $(1+\varepsilon)$-approximation to the minimum ℓ'-cost path subject to the bounds $\mathcal{B}_j$ on the r' costs and $\mathcal{D}$ on the d-weights between the centers v_x and v_y in G'.
      ii. 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 ℓ'-cost of $P_{xy}(\mathcal{B}_j)$. Also, the r'-cost of the edge is set to that of $P_{xy}(\mathcal{B}_j)$.
   (d) Find a r'-cost $(1+1/\varepsilon)(1+\varepsilon)\mathcal{B}$ bounded $(1+\varepsilon)$-approximate minimum ℓ'-weight matching of largest cardinality in G_i.
   (e) For each edge $e = (v_x, v_y)$ in the matching, merge the clusters C_x and C_y, for which v_x and v_y were centers respectively, by adding the corresponding 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 the cluster C_{xy}. C_{xy} is added to the cluster set $\mathcal{C}_{i+1}$ for the next phase, while C_x and C_y are removed.
6. Let $\hat{C}$, with center $\hat{v}$ be the single cluster left after Step 5. Compute a shortest path tree T' of $\hat{C}$ rooted at $\hat{v}$ using the d-weights.
7. For each multiedge $e' = (u, v) \in T'$ define the reduction r on the corresponding edge $e = (u, v)$ in the original graph by $r(e) := r'(e')$. Define a tree T in G by including the corresponding edges e. Output r and T.

Output: A reduction r and a Steiner tree T in G of total length, with respect to the modified length function $\ell - r$, at most $(1+\varepsilon)^3 \lceil \log_2 |K| \rceil$ times that of the optimally improved Steiner tree. The d-diameter of T is at most $2\lceil \log_2 |K| \rceil \mathcal{D}$ and the cost of the reduction r is bounded from above by $(1+\varepsilon)(1+1/\varepsilon)\lceil \log_2 |K| \rceil \mathcal{B}$.

Figure 2: Approximation algorithm for Dia-BCST.

in Improve-Dia-BCST can be easily seen to have running times independent of the weights. We employ Hassin's strongly polynomial FPAS for Step 5(c)i [18]. Hassin's approximation algorithm runs in time $O(|E|(\frac{n^2}{\epsilon}\log\frac{n}{\epsilon}))$. Thus, Improve-Dia-BCST is a strongly polynomial time algorithm. The construction of an approximation to the constrained matching in Step 5d can be implemented by using the parametric search technique employed in [27] for finding a $(1+1/\varepsilon, 1+\varepsilon)$ approximation algorithm for bicriteria problems with similar costs.

4.1 Correctness and Performance Guarantee

Assume in the following that r^* is an optimal reduction strategy for the graph G and that T^* is a corresponding optimal constrained Steiner tree of total weight $\mathrm{OPT} := \mathrm{Steiner}_G(\ell - r^*, d, \mathcal{D})$. We first state two basic lemmas for graphs with multiple weights on the edges. The first can be proved by an extension of Hassin's work [18]:

Lemma 4.1 If there are three cost functions ℓ', r' and d and there are bounds $\mathcal{B}'$ on the r' costs and $\mathcal{D}$ on d costs respectively, there is a polynomial time algorithm that finds a path between any two specified vertices such that it obeys the constraints on the r' and d costs and is within $(1+\varepsilon)$ times the optimal shortest constrained ℓ'-path between the two vertices, for any fixed $\varepsilon > 0$.

The result of the second lemma can be established by using a parametric search technique similar to the one devised in [27] in conjunction with well known polynomial time minimum weight matching algorithms (cf. [7]):

Lemma 4.2 Let $\varepsilon > 0$ be fixed. Given a complete graph with two cost functions ℓ' and r' and a bound $\mathcal{B}'$, there is a polynomial time algorithm which finds a matching in the graph where the total r'-cost of the edges in the matching is at most $(1+1/\varepsilon)\mathcal{B}'$ and of ℓ'-cost at most $(1+\varepsilon)$ times that of an optimal constrained matching.

We now show that the transformation procedure does not destroy important properties with respect to the problem.

Lemma 4.3 The graph G' obtained as a result of applying procedure Transform to G satisfies the following properties:

1. There exists a Steiner tree T' in G' of total ℓ'-length at most OPT, r'-cost no more than $(1+\varepsilon)\mathcal{B}$ and of d-diameter at most $\mathcal{D}$.
2. Let $u, v \in K$ be two terminals. Let P be the unique path between u and v in the optimally improved tree T^*. Let $(\ell - r^*)(P)$ and cost_P denote the length with respect to $\ell - r^*$ and the money spent by r^* on the edges of P in T^* respectively. Then, in G' there exists a path between u and v of d-diameter at most $\mathcal{D}$, ℓ'-length at most $(\ell - r^*)(P)$ and r'-cost at most $(1+\varepsilon)\mathrm{cost}_P$.

Proof: We only show the first part of the lemma. The second part can be proved similarly.

Consider the optimal tree T^* in G. We can define a tree T' in G' in the following way: For an edge $e = (u, v) \in T^*$ that is reduced by $r^*(e)$ we select a multiedge $e' = (u, v)$ in G' of ℓ'-cost $\ell(e) - b(e)$, where $b(e)$ is selected in such a way that $\frac{b(e)}{(1+\varepsilon)} \le r^*(e) \le b(e)$. Observe that the edge e' selected in the above fashion has (modified) length at most $\ell(e) - r^*(e)$ and at least $\frac{\ell(e)-r^*(e)}{1+\varepsilon}$.

Using the above facts on the cost values associated with e', it follows that the total ℓ'-length of T' is at most that of T^*. Also, the r'-cost of T' is at most the cost of the optimal reduction r^* multiplied by $(1+\varepsilon)$, and thus bounded from above by $(1+\varepsilon)\mathcal{B}$. It is straightforward to see that the d-diameters of T' and T^* coincide. $\square$

Observation 4.4 The set of cluster centers maintained by the algorithm is always a subset of the terminals K of the original graph G.

Claim 4.5 The Algorithm Improve-Dia-BCST 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 5e). Hence we are left with one cluster after phase $\lceil \log_2 |K| \rceil$ and Algorithm Improve-Dia-BCST terminates. $\square$

Lemma 4.6 ([8, 29]) Let T be a tree with an even number of marked nodes. Then there is a pairing $(v_1, w_1), \ldots, (v_k, w_k)$ of the marked nodes such that the $v_i - w_i$ paths in T are edge-disjoint.

Claim 4.7 Let $C \in \mathcal{C}_i$ be any cluster in phase i of Algorithm Improve-Dia-BCSTand let v be its center. Then any node u in C is reachable from v by a path in C is of d-weight at most iD.

Proof: By Observation 4.4, the set of cluster centers in phase i is a subset of the terminals K. By Lemma 4.3, there is a Steiner tree in G' of d-diameter at most $\mathcal{D}$ and of total r'-cost no more than $(1+\varepsilon)\mathcal{B}$ (which is an upper bound on $\mathcal{B}_j$ for all j). This implies that all the paths $P_{xy}(\mathcal{B}_j)$ which are constructed in Step 5(c)i with the help of Hassin's restricted shortest path algorithm have d-weight at most $\mathcal{D}$. The claim of the lemma now follows by a straightforward induction on i. $\square$

Using the result of the last claim in conjunction with Claim 4.5 we can establish the following.

Lemma 4.8 Algorithm Improve-Dia-BCST outputs a Steiner tree T with d-diameter at most $2\lceil \log_2 |K| \rceil \cdot \mathcal{D}$.

Proof: It follows from Claims 4.7 and 4.5 that the tree T' found in Step 6 has d-diameter at most $2\lceil \log_2 |K| \rceil \cdot \mathcal{D}$. By construction of r and T in Step 7, the d-diameter of T in G is no more than the d-diameter of T' in G'. $\square$

This completes the proof of the performance guarantee with respect to the d-cost. We now proceed to prove the performance guarantee with respect to the modified ℓ-lengths.

Claim 4.9 Let $\hat{T}$ be any minimum ℓ'-cost Steiner tree in G' subject to the constraint that the d-diameter does not exceed $\mathcal{D}$ and its total r'-cost is bounded by $(1+\varepsilon)\mathcal{B}$. The ℓ'-weight of the constrained minimum weight matching found in Step 5e in any phase of the Algorithm Improve-Dia-BCST is at most $(1+\varepsilon)^3 \cdot \ell'(\hat{T})$.

Proof: Notice that by Lemma 4.3 a tree satisfying both the constraints as stated in the claim does exist.

Let i be fixed and consider the i'th phase of the algorithm. By Observation 4.4 all the centers $v_1, v_2, \ldots, v_{k_i}$ in the i'th stage are terminals. Thus, all these vertices are contained in the Steiner tree $\hat{T}$. We now mark the vertices $v_1, v_2, \ldots, v_{2\lfloor \frac{k_i}{2} \rfloor}$ in $\hat{T}$ and apply Lemma 4.6 to $\hat{T}$ with these vertices marked. The lemma states that there is a pairing of the centers, which we can assume without loss of generality to be $(v_1, v_2), \ldots, (v_{2\lfloor\frac{k_i}{2}\rfloor-1}, v_{2\lfloor\frac{k_i}{2}\rfloor})$, such that the paths between the paired vertices in $\hat{T}$ are edge disjoint.

By the disjointness of these paths, their total ℓ'-cost is at most $\ell'(\hat{T})$. Similarly, the total r'-cost of the edges in these paths does not exceed $(1+\varepsilon)\mathcal{B}$, and none of the paths has length more than $\mathcal{D}$ under the d-weight function.

By construction of the paths $P_{xy}(\mathcal{B}_j)$ in Step 5(c)i of the algorithm, we can find for each of the paths P between the centers from above, a path P' among the paths $P_{xy}(\mathcal{B}_j)$ in G' with the following properties: (i) The total r'- and d-lengths of P' are no more than the corresponding values for P. (ii) The ℓ'-cost of P' is at most $(1+\varepsilon)^2$ times that of P. Here, one factor $(1+\varepsilon)$ comes from the approximation via the constrained shortest path algorithm, while the second stems from the discrete budget values $\mathcal{B}_j$, which are used for the constrained paths.

Using the edges in G_i corresponding to this selection from the paths $P_{xy}(\mathcal{B}_j)$ yields a matching of the vertices $(v_{2\lfloor\frac{k_i}{2}\rfloor-1}, v_{2\lfloor\frac{k_i}{2}\rfloor})$ in G_i of ℓ'-cost at most $(1+\varepsilon)^2\ell'(\hat{T})$ and of r'-cost at most $(1+\varepsilon)\mathcal{B}$.

Since in Step 5d of the Algorithm Improve-Dia-BCST, a matching of r'-cost at most $(1+1/\varepsilon)(1+\varepsilon)\mathcal{B}$ is found that is of total ℓ'-weight at most $(1+\varepsilon)$ that of the minimum ℓ'-cost and r'-cost $(1+\varepsilon)\mathcal{B}$ bounded matching, the claim follows. $\square$

We are now ready to estimate the ℓ'-length of the Steiner tree T.

Lemma 4.10 The Algorithm Improve-Dia-BCSToutputs a Steiner tree T with total length under the modified length function $\ell - r$ of at most $(1+\varepsilon)^3\lceil\log_2|K|\rceil \cdot \mathrm{OPT}$.

Proof: By the results of Claims 4.5 and 4.9, the total ℓ'-length of the edges in the single cluster $\hat{C}$ is at most $(1+\varepsilon)^3\lceil\log_2|K|\rceil \cdot \ell'(\hat{T})$, where $\hat{T}$ is an optimal constrained Steiner tree in G'. By Lemma 4.3, $\ell'(\hat{T}) \leq \mathrm{OPT}$. Thus, the total ℓ'-length of the shortest path tree T, which is computed in Step 6 is no more than $2(1+\varepsilon)^3\lceil\log_2|K|\rceil \cdot \mathrm{OPT}$. By the construction of the reduction r and the tree T in the final Step 7, the length of T under the modified length function $\ell - r$ is bounded from above by $\ell'(T')$. This establishes the lemma. $\square$

We now address the r'-cost of the edges that are added during an arbitrary phase of the algorithm. Using the argument from the proof of Claim 4.9, we see that a matching of total r'-cost at most $(1+1/\varepsilon)(1+\varepsilon)\mathcal{B}$ can be found in Step 5d. This leads to the following.

Claim 4.11 The r'-cost of the edges added in Step 5e in any phase of Algorithm Improve-Dia-BCST is at most $(1+\varepsilon)(1+1/\varepsilon)\cdot\mathcal{B}$.

Now, using again the result of Lemma 4.5 that there are at most $\lceil\log_2|K|\rceil$ phases during the run of the algorithm, the following lemma is now evident:

Lemma 4.12 The total r'-cost of the tree T' found in Step 6 is at most $(1+\varepsilon)(1+1/\varepsilon)\lceil\log_2|K|\rceil \cdot \mathcal{B}$. Moreover, the cost of the reduction r constructed in Step 7 is also bounded by $(1+\varepsilon)(1+1/\varepsilon)\lceil\log_2|K|\rceil \cdot \mathcal{B}$.

Combining the results of Lemmas 4.8, 4.10 and 4.12, we obtain Theorem 4.13.

Theorem 4.13 Algorithm Improve-Dia-BCST is an approximation algorithm for Dia-BCST, which for any fixed $\varepsilon > 0$ performs as follows: Given any instance of the Dia-BCSTproblem it outputs a reduction r and a Steiner tree T such that:

1. The length of T with respect to the modified lengths in G satisfies:

$$(\ell - r)(T) \leq (1+\varepsilon)^3 \lceil \log_2 |K| \rceil \cdot \mathrm{OPT}.$$

2. The d-diameter of T does not exceed $2\lceil \log_2 |K| \rceil \cdot \mathcal{D}$.
3. The cost of the reduction r is at most $(1+\varepsilon)(1+1/\varepsilon)\lceil \log_2 |K| \rceil \cdot \mathcal{B}$.

5 Approximation Algorithm for Deg-BCST

In this section, we present our approximation algorithm for obtaining a near optimal degree bounded minimum cost Steiner tree. We call this problem Deg-BCST. The basic idea is very similar to the Deg-BMST problem and so we only highlight the details. As in the case of the approximation of Deg-BMST, we first transform the input graph G to another graph G' using the transformation procedure from Figure 1[6]. After this step, our algorithm for Deg-BMST works in *phases*. In each phase, we add a subgraph which approximately maintains the degree bound has a good total cost and does not cost too much to upgrade it. For determining the edges of the subgraph to be added, we will again use a variant of the matching algorithm. The matching algorithm used is for a version of the DCS problem, where we use again a two cost version of the problem. Since the ordinary DCS problem can be solved in polynomial time, we can use a parametric search technique to reduce the problem of two costs to one cost.

5.1 Background

In this section, we provide some background material on a degree constrained subgraph (DCS) problem. The general DCS problem can be stated as follows: Given an undirected graph with nonnegative costs on the edges, and an integer-valued function f defined on the vertices of the graph, find a minimum-cost subgraph (if one exists) such that the degree of any vertex v in this subgraph is $f(v)$. This problem is referred to as the *f-factor problem* [7], and is known to be polynomially solvable [6, 7]. By using matching techniques given in [7] one can extend this result:

Fact 5.1 (b-bounded even DCS problem, [7]) The following problem has a polynomial-time solution: Given an undirected graph $G=(V,E)$ such that $V = S \cup U$ and $S \cap U = \emptyset$, and an integer $b \geq 2$, find a subgraph H (if one exists) of G of minimum cost such that

- for all vertices $v \in U$, we have $\deg_H(v) = 1$, and
- for all vertices $v \in S$, we have $0 \leq \deg_H(v) \leq b$ and $\deg_H(v) \equiv 0 \pmod 2$.

Here, $\deg_H(v)$ denotes the degree of the node v in the subgraph H.

The b-bounded even DCS problem described above is a generalization of the T-join problem [6]. When b is allowed to be unbounded, this problem reduces to the T-join problem for the set T. We will use the above results in the proof of the performance guarantee of our approximation algorithm. The following result can be obtained by combining Fact 5.1 with a parametric search technique similar to the one in [27]:

Lemma 5.2 Let $\varepsilon > 0$ be fixed. Given a graph with two cost functions ℓ' and r', a bound b on the degree and a bound $\mathcal{B}'$, there is a polynomial time algorithm which finds a DCS H with total

[6]Actually, the transformation procedure used here is a trivial modification of Transform, where no d-weights are involved.

r'-cost of the edges in the subgraph at most $(1+1/\varepsilon)\mathcal{B}'$ and of ℓ'-cost at most $(1+\varepsilon)$ times that of an optimal constrained DCS.

Similarly, using the same parametric search technique in conjunction with the approximation algorithm for the Steiner tree problem given in [17], the following lemma can be established.

Lemma 5.3 Let $\varepsilon > 0$ be fixed and let a graph with terminals $K \subseteq V$, two cost functions ℓ' and r' on the edges and a bound $\mathcal{B}'$ on its r'-cost be given. Then there is a polynomial time algorithm which finds a Steiner tree with total r'-cost of the edges in the subgraph at most $2(1+1/\varepsilon)\mathcal{B}'$ and of ℓ'-cost at most $2(1+\varepsilon)$ times that of an optimal constrained Steiner tree.

We need one more result on the structure of degree constrained subgraphs:

Proposition 5.4 Let H be any subgraph satisfying the conditions in Fact 5.1. Then there is a pairing of the nodes of U in H such that there are edge-disjoint paths in H between each pair of vertices.

5.2 Approximation Algorithm Outline and Performance Guarantee

We now describe our approximation algorithm for Deg-BMST. The algorithm follows the same skeletal outline as an early algorithm [9] of Fürer and Raghavachari for approximating the minimum-degree spanning tree. However, we generalize it to Steiner trees and ensure that the cost of the solution chosen is small as well.

As in the case of the approximation of Dia-BCST, we first transform the input graph G to another graph G' using the transformation procedure from Figure 1. After this step, our algorithm for Deg-BMST works in *phases*. There are $\mathcal{O}(\log \frac{|K|}{b})$ phases. At the beginning, the solution subgraph is empty and each terminal is in a connected component by itself. We call a connected component *active*, if it contains at least one terminal. At each iteration, we add edges between the components, reducing the number of active components in the current solution by at least a factor of half. When the number of active components falls to $\mathcal{O}(b)$, we compute a (constrained) minimum Steiner tree in the current graph to connect up all the remaining active components.

The choice of edges to include in each phase will be done in the following way: (1) The reduction cost spent on the edges is no more than the given budget $\mathcal{B}$. (2) The degree of any node in the graph increases by at most $\mathcal{O}(b)$. (3) Finally, the solution cost with respect to the modified ℓ costs is bounded from above by OPT_b.

Using these properties, we can establish the performance guarantee. The complete specification of Algorithm Improve-Deg-BCST is shown in Figure 3 on page 12.

In each iteration, the solution to the DCS problem may contain one or more copies of an edge $e \in E'$. In any case, in Step 4d, we include only one copy of such an edge in the set F. The results on the performance are summarized in the following theorem. A proof sketch can be found in Section B of the appendix.

Theorem 5.5 Algorithm Improve-Deg-BCST is a polynomial-time approximation algorithm for Deg-BMST, which for any fixed $\varepsilon > 0$ performs as follows: Given any instance of the Deg-BMSTproblem, the algorithm finds a reduction r and a Steiner tree T in G such that:

1. The length of T with respect to the modified lengths $\ell - r$ in G satisfies:

$$(\ell - r)(T) \leq (1+\varepsilon)^2 \lceil \log_2 \frac{|K|}{b} \rceil \cdot \mathrm{OPT}_b = \mathcal{O}(\log \frac{|K|}{b}) \cdot \mathrm{OPT}_b.$$

2. The maximum degree of T is $\mathcal{O}(\log \frac{|K|}{b}) \cdot b$.
3. The cost of the reduction r is bounded by $(1+\varepsilon)(1+1/\varepsilon)\lceil \log_2 \frac{|K|}{b} \rceil \mathcal{B} = \mathcal{O}(\log \frac{|K|}{b}) \cdot \mathcal{B}$

6 Hardness Results

In this section we briefly mention results on the hardness and non-approximability of the Dia-BCST and Deg-BCST problems. By a reduction from the Continuous Multiple Choice Knapsack problem (cf. [15, MP11] for a definition) similar to the one in [20] we can also show the hardness of Dia-BCST for very restricted classes of graphs:

Theorem 6.1 Dia-BCST is NP-hard, even when restricted to series-parallel graphs, the d-costs are all equal to zero and the set of terminals equals the vertex set of the input graph.

The next non-approximability result can be proved by using an approximation preserving reduction from Set Cover in conjunction with the hardness results from [26, 2]:

Theorem 6.2 Unless NP $\subseteq$ DTIME$(n^{\log\log n})$, for any fixed $\alpha < \frac{1}{8}\log|K|$, $\beta \geq 1$, there can be no polynomial time approximation algorithm for Dia-BCST, which always finds a reduction of cost at most $\beta\mathcal{B}$ and a Steiner tree of diameter at most $\mathcal{D}$ whose length in the modified graph is at most α times the optimally improved tree. The result continues to hold, if we replace β by any polynomial time computable function and the set of terminals equals the set of vertices, i.e. if we look for a spanning tree of the input graph.

Finally, using a reduction from the Hamiltonian path problem, we obtain:

Theorem 6.3 For any $\alpha, \beta \geq 1$ the following statement holds: If there is a polynomial time approximation which, given any instance of Deg-BCST, finds a reduction of cost at most $\beta\mathcal{B}$ and a Steiner tree T of bounded degree b such that the length $(\ell - r)(T)$ in the modified graph is at most α times the optimally improved tree, then P $=$ NP. This result continues to hold if α, β are replaced by polynomial time computable functions and the set K of terminal equals V.

7 Concluding Remarks

We have presented the first polynomial time approximation algorithms with guaranteed performances for network improvement problems with multiple constraints. Several of our results can be extended to the more general one connected subgraph problems.

Algorithm Improve-Degree-Constrained-Tree (Improve-Deg-BCST):

Input: $G = (V, E)$ - Graph with three edge cost functions, ℓ, $\ell_{\min}$ and c
b - Bound on the degree of the tree
$\mathcal{B}$ - Bound on the amount of money spent in updating the graph
$K \subseteq V$ - Set of terminals, and
ε - Accuracy parameter.

1. Call $\mathsf{Transform}(\varepsilon)$ to obtain a new graph $G = (V, E')$ with two weights $\ell'(e)$, $r'(e)$ on the edges $e \in E'$.
2. Initialize the set of edges in the solution subgraph $F := \emptyset$, and the iteration count $i := 1$.
3. Initialize the set of active components to contain $|K|$ singleton sets, one for each terminal in K.
4. Repeat until the number of active components is $\mathcal{O}(b)$
   (a) Let $\mathcal{C}_i = \{C_1 \dots, C_{p_i}\}$ be the set of active components of (V, F).
   (b) Construct an auxiliary graph G_i as follows.
      i. The node set of G_i is $V \cup \mathcal{C}_i$.
      ii. We now describe the edges included in G_i:
      Between all the nodes of V in each active component $C_j \in \mathcal{C}_i$, include edges of zero ℓ'- and r'-cost to form a clique in G_i.
      Let $\tilde{E} \subseteq E'$ represent the set of edges of G whose endpoints are in different components in $\mathcal{C}_i$. For each edge $e = (u, v) \in \tilde{E}$ of length $\ell'(e)$ and cost $r'(e)$, include four sets of edges in G_i: let $u \in C_u$ and $v \in C_v$. We include the sets of edges $(u, v), (u, C_v), (v, C_u)$ and (C_u, C_v), each of ℓ'-length $\ell'(e)$ and r'-cost $r'(e)$, in G_i.
      iii. If $|\mathcal{C}_i|$ is odd, we include an extra dummy node z in $\mathcal{C}_i$ and include zero-cost edges between z and every $C_j \in \mathcal{C}_i$ in G_i.
   (c) Find a r'-cost $(1+1/\varepsilon)(1+\varepsilon)\mathcal{B}$ bounded $(1+\varepsilon)$-approximate minimum ℓ'-weight $2b$-bounded even DCS of G_i with nodes in V having even degree (between 0 and $2b$) and the nodes in $\mathcal{C}_i$ having degree one.
   (d) Add the edges from G' found by the DCS-algorithm in Step 4c to F.
   (e) Update the set of clusters and set $i := i + 1$.
5. The number of active components of (V, F) is now $\mathcal{O}(b)$. Contract each active component of (V, F) to a single node.
6. In the resulting graph, use the active components as terminals and compute a Steiner tree of r'-cost at most $2(1 + 1/\varepsilon)(1 + \varepsilon)\mathcal{B}$ and of ℓ'-weight at most $2(1 + \varepsilon)$ times that of an minimum ℓ'-weight Steiner tree subject to the constraint that its r'-cost does not exceed $(1 + \varepsilon)\mathcal{B}$.
   Add the edges of this Steiner tree to the set F.
7. Find a spanning tree T' of (V, F). For each edge $e' = (u, v) \in T'$ define the reduction r on the corresponding edge e in the original graph by $r(e) := r'(e')$. Define a tree T in G by including the corresponding edges e. Output r and T.

Output: A reduction r and Steiner tree T in G of total length, with respect to the modified length function $\ell - r$, at most $(1 + \varepsilon)^2 \lceil \log_2 n \rceil$ times that of the optimally improved Steiner tree subject to the degree constraint. The degree of T is $\mathcal{O}(b \log \frac{|K|}{b})$ and the cost of the reduction r is bounded from above by $(1 + \varepsilon)(1 + 1/\varepsilon)\lceil \log_2 n \rceil \mathcal{B}$.

Figure 3: Approximation algorithm for $\mathsf{Deg\text{-}BMST}$.

References

[1] A. Agrawal, P. Klein and R. Ravi, "When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks", *SIAM J. Comput.*, Vol. 24, No. 3, pp. 440-456, June 1995.

[2] M. Bellare, S. Goldwasser, C. Lund and A. Russel, "Efficient Probabilistically Checkable Proofs", Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, 1993, pp. 294–304.

[3] O. Berman, "Improving The Location of Minisum Facilities Through Network Modification," *Annals of Operations Research*, 40(1992), pp. 1–16.

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

[5] H.L. Bodlaender, "Dynamic programming algorithms on graphs with bounded treewidth", Proceedings of the 15th International Colloquium on Automata, Languages and Programming, Springer-Verlag, LNCS 317, 1988, pp. 105–119

[6] J. Edmonds, and E. L. Johnson, "Matching, Euler tours and the Chinese postman", *Math. Prog. 5*, (1973), pp. 88-124.

[7] L. Lovász and M. D. Plummer, *Matching theory*, Akadémiai Kiadó, Budapest (1986).

[8] P. 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* (1993), pp. 323-332.

[9] M. Fürer and B. Raghavachari, "An $\mathcal{NC}$ approximation algorithm for the minimum degree spanning tree problem," in *Proceedings of the 28th Annual Allerton Conference on Communication, Control and Computing* (1990), pp. 274-281.

[10] J. Cong, A. B. Kahng, G. Robins, M. Sarafzadeh and C. K. Wong, "Provably Good Performance Driven Global Routing," *IEEE Transactions on Computer Aided Design*, 11(6), 1992, pp. 739-752.

[11] J. P. Cohoon and L. J. Randall, "Critical Net Routing," *IEEE Intern. Conf. on Computer Design,* 1991, pp. 174-177.

[12] W. Cunningham, "Optimal Attack and Reinforcement of a Network," *J. ACM*, 32(3), 1985, pp. 549-561.

[13] G.N. Frederickson and R. Solis-Oba, "Increasing the Weight of Minimum Spanning Trees", *Proceedings of the Sixth Annual ACM-SIAM SODA'96*, March 1996.

[14] H. N. Gabow, Z. Galil, T. H. Spencer and R. E. Tarjan, "Efficient Algorithms for Finding Minimum Spanning Trees in Undirected and Directed Graphs," *Combinatorica*, 6 (1986), pp. 109–122.

[15] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.

[16] M. X. Goemans, and D. P. Williamson, "A General Approximation Technique for Constrained Forest Problems", *Proceedings of the Third Annual ACM-SIAM SODA'92*, Jan. 1992, pp. 307-316.

[17] 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 SODA'94*, Jan. 1994, pp. 223-232.

[18] R. Hassin, "Approximation Schemes for the Restricted Shortest Path Problem," *Math. of OR*, 17(1), 1992, pp. 36-42.

[19] J. Ho, D.T. Lee, C.H. Chang and C.K. Wong, "Bounded Diameter Spanning Tree and Related Problems," *Proc. of the Annual ACM Symposium on Computational Geometry,* 1989, pp. 276-282.

[20] S. O. Krumke, H. Noltemeier, M. V. Marathe, S. S. Ravi and K. U. Drangmeister, "Modifying Networks to Obtain Low Cost Trees," submitted for publication.

[21] D. Karger and S. Plotkin, "Adding Multiple Cost Constraints to Combinatorial Optimization Problems, with Applications to Multicommodity Flows," *Proc. 27th Annual ACM Symp. on Theory of Computing* (STOC'95), May 1995, pp. 18–25.

[22] B. Kadaba and J. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Trans. on Communication*, Vol. COM-31, Mar. 1983, pp. 343-351.

[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*, 1993, pp. 286–292.

[26] C. Lund and M. Yannakakis, "On the Hardness of Approximating Minimization Problems," *Proc., 25th Annual ACM Symp. on Theory of Computing* (STOC'93), May 1993, pp. 288–293.

[27] M. V. Marathe, R. Ravi, S. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III, "Bicriteria Network Design Problems", *Proc. ICALP'95*, July 1995.

[28] C. Phillips, "The Network Inhibition Problem," *Proc. 25th Annual ACM STOC'93*, May 1993, pp. 288–293.

[29] 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," *Proc. 25th Annual ACM STOC'93*, May 1993, pp. 438–447.

[30] R. Ravi, "Rapid Rumor Ramification: Approximating the Minimum Broadcast Time," *Proceedings of the 35th Annual FOCS'94*, Nov. 1994, pp. 202–213.

[31] A. Warburton, "Approximation of Pareto optima in Multiple–Objective, Shortest Path Problems," *Operations Research*, Vol. 35, 1987, pp. 70–79.

[32] Q. Zhu, M. Parsa and W. Dai, "An Iterative Approach for Delay Bounded Minimum Steiner Tree Construction," Technical Report UCSC-CRL-94-39, University of California, Santa Cruz, Oct 1994.

Appendix

A Notes on Algorithm Improve-Dia-BCST

In Algorithm Improve-Dia-BCST presented in Section 4 the transformation procedure plays an important role. Recall that the algorithm added multiedges that reflected discrete values of improvement on a particular edge. We now show how to avoid multiedges by choosing a certain subgraph to replace an edge. The modified transformation procedure is shown in Figure 4 and an example of the transformation is depicted in Figure 5. The results on the performance of Algorithm Improve-Dia-BCST continue to hold, if we use the new transformation procedure. An advantage of the modified transformation is that it does not cause the treewidth of the graph to increase. We use this fact to obtain improved approximation algorithms for the class of treewidth bounded graphs.

Procedure Transform2(ε)
1 **for** each edge $e = (u, v)$ in the graph let b_e be chosen so that $(1+\varepsilon)^{b_e} \leq \ell(e) - \ell_{\min}(e) \leq (1+\varepsilon)^{b_e+1}$.
2 Add $b_e + 2$ new vertices r_k, $k = -1, 0, \ldots, b_e$, which are joined together in a simple cycle.
3 **for all** k, $-1 \leq k \leq b_e$, join r_k to both u and v.
4 For $k \geq 0$, the edge (u, r_k) has length $\ell'(u, r_k) = \ell(e) - (1+\varepsilon)^k$, cost $r'(u, r_k) = (1+\varepsilon)^k c(e)$ and d-weight $d(u, r_k) := d(e)$, while the edge (u, r_{-1}) has length $\ell(e)$, cost 0 and weight $d(e)$. All the edges (r_k, v) and (r_k, r_{k+1}) are of zero length, cost and weight.
(An example of the above transformation for $b_e = 2$ can be seen in Figure 5).

Figure 4: Transformation procedure avoiding multiedges.

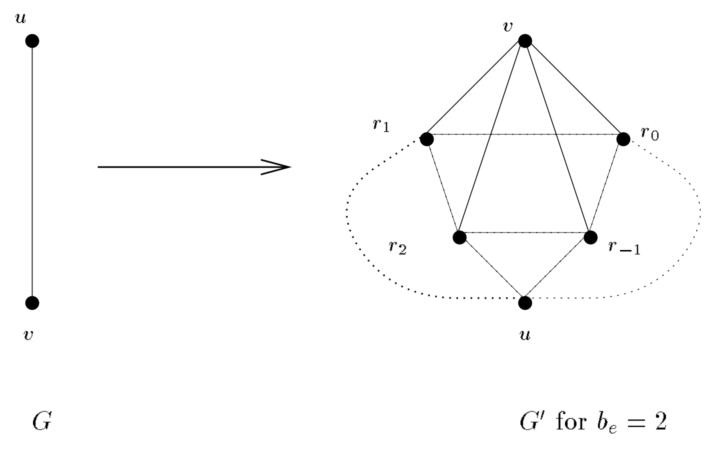

Figure 5: Example for the modified transformation.

We can use a modification of Algorithm Improve-Dia-BCST in conjunction with a binary search to obtain an algorithm for a related problem with *one* measure for the tree. Given a graph with lengths $\ell(e)$ (and minimum lengths $\ell_{\min}(e)$) and costs $c(e)$ ($e \in E$), the *BMDST problem* is to find a reduction strategy r^* of cost at most $\mathcal{B}$ such that the diameter of a minimum diameter Steiner tree with respect to $\ell - r^*$ is minimized. The algorithm then finds a reduction r^* of cost $\mathcal{O}(\log n) \cdot \mathcal{B}$ such that the diameter of the minimum diameter Steiner tree in the modified network is within $\mathcal{O}(\log n) \cdot \mathrm{OPT}_{\mathrm{dia}}$, where $\mathrm{OPT}_{\mathrm{dia}}$ denotes the diameter of a tree resulting from an optimal reduction. A brief sketch of how the modification is done is discussed in the sequel.

The input for the *Budget Minimum Diameter Steiner Tree Problem* is given by an undirected graph G and nonnegative edge-weight functions $\ell, \ell_{\min}, c$, which have the same meaning as for Deg-BMST. Again, we are given a budget $\mathcal{B}$ that is available for reducing the lengths of the edges. Denote by $\mathrm{MDST}_G(\ell)$ the diameter of a minimum diameter Steiner tree of G with respect to the lengths given by ℓ. The problem can then be defined as follows:

Definition A.1 The *Budget Minimum Diameter Steiner Tree Problem (BMDST)* is to find a feasible reduction r such that $\mathrm{MDST}_G(\ell - r)$ has the least possible value.

Using a reduction from Set Cover, it can be shown that BMDST is NP-hard, even if $K = V$; i.e., if we seek for a minimum spanning tree instead of a more general Steiner tree and $c(e) = 1$ for all $e \in E$. With the help of the non-approximability results for Set Cover from [26] we can even deduce that the problem remains hard to solve[7], if we allow a violation of the budget-constraint by a factor of $\mu \log_2 B$ for any fixed μ, $0 < \mu < 1/8$.

We first modify Algorithm Improve-Dia-BCST to a *relaxed decision procedure*, which gives information whether there is a Steiner tree of diameter at most $\mathcal{D}$ if a budget of $\mathcal{B}$ is spent on the reduction. The decision will be relaxed in the sense that for a given value of $\mathcal{D}$ the algorithm does one of the following:

1. The algorithm tells us that the diameter of $\mathcal{D}$ cannot be achieved with a budget of $(1+\varepsilon)\mathcal{B}$.

2. The algorithm delivers a reduction of cost $\mathcal{O}(\mathcal{B} \log n)$ and a Steiner tree of diameter $\mathcal{O}(\mathcal{D} \log n)$.

Observe that the optimum diameter achievable with a budget of $\mathcal{B}$ is contained in the interval $\mathcal{I} := [(n-1) \min_{e \in E} \ell_{\min}(e), (n-1) \max_{e \in E} \ell_{\min}(e)]$. After the modification to Improve-Dia-BCST, we can perform a binary search to find the minimum value in $\mathcal{I}$ such that the algorithm outputs a Steiner tree. It can be shown that we obtain a logarithmic performance while violating the budget-constraint by a factor of at most $\mathcal{O}(\log n)$. We briefly mention the main modifications to Improve-Dia-BCST:

1. The values $\mathcal{B}_j$ do not have to be computed.

2. In Step 5(c)i we calculate paths P_{xy} which are computed to be $(1+\varepsilon)$-approximations to the minimum r'-cost paths subject to the bound $\mathcal{D}$ on their d-length.

3. In Step 5e we compute a minimum total r'-cost perfect matching.

This enables us to show the following.

Theorem A.2 There is a polynomial time approximation algorithm for BMDST, which for any fixed $\varepsilon > 0$ performs as follows: Given any instance of the BMDST problem, it outputs a reduction r and a Steiner tree T such that:

[7] The problem cannot be solved in polynomial time unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log \log n})$.

1. The cost of the reduction r is $\mathcal{O}(\log n) \cdot \mathcal{B}$.

2. The diameter of T with respect to the modified lengths $\ell - r$ in G satisfies:

$$\mathrm{dia}_{(\ell-r)}(T) \leq \mathcal{O}(\log n) \cdot \mathrm{MDST}_G(\ell - r^*),$$

where r^* is an optimal reduction strategy for a budget of $\mathcal{B}$.

□

B Proof of Theorem 5.5

Throughout the analysis we adopt the following notation: We use b to denote the degree bound specified in the problem. We let r^* be an optimal reduction strategy, T^* a corresponding minimum Steiner tree of bounded degree and $\mathrm{OPT}_b = \mathrm{Steiner}_G(\ell - r^*, \deg(v), b)$ denote its total weight in the modified graph.

First we have to show that during the transformation of the graph basically no structure is lost. The following lemma can be proved in a manner similar to Lemma 4.3:

Lemma B.1 Let G' be the graph resulting from the transformation. Then the following statements hold:

1. There exists a Steiner tree T' in G' of total ℓ'-length at most OPT_b, r'-cost no more than $(1+\varepsilon)\mathcal{B}$ and such that each node in T' has degree no more than b.

2. Let $u, v \in V$ be two vertices. Let P be the unique path between u and v in the optimally improved tree T^*. Let $(\ell - r^*)(P)$ and cost_P denote the length with respect to $\ell - r^*$ and the money spent by r^* on the edges of P in T^* respectively. Then, in G' there exists a path between u and v of ℓ'-length at most $(\ell - r^*)(P)$ and r'-cost at most $(1+\varepsilon)\mathrm{cost}_P$.

At each iteration, applying Proposition 5.4 to the solution for the DCS problem in Step 4c, we can derive a pairing of the active components in $\mathcal{C}$ such that there are paths between these pairs in the solution. This observation can be used to show that in each phase the number of active components of (V, F) reduces by at least a factor of half. Thus, there are $\mathcal{O}(\log \frac{|K|}{b})$ iterations. By the constraint on the degree of the nodes in the subgraph added at each iteration of the algorithm, the degree of any node increases only by $\mathcal{O}(b)$ in each step. Observe also that the edges added to the solution resulting from the Steiner tree computed in Step 7 will add only $\mathcal{O}(b)$ to the degree. The following lemma is now almost immediate:

Lemma B.2 The total number of iterations of Algorithm Improve-Deg-BCST is $\lceil \log_2 \frac{|K|}{b} \rceil = \mathcal{O}(\log \frac{|K|}{b})$. The degree of the spanning tree output by the above algorithm is $\mathcal{O}(b \log \frac{|K|}{b})$.

Lemma B.3 The total ℓ'-length of the edges added to the solution in Step 6 is at most $2(1+\varepsilon)\mathrm{OPT}_b$.

Proof: Contract each active component of (V, F) to a single supernode. Also, consider the $\mathcal{O}(b)$ active components as terminals. A minimum ℓ'-weight Steiner tree $\tilde{T}$ in G' subject to the bounds b on its degree and $(1+\varepsilon)\mathcal{B}$ on its r'-cost, induces a Steiner tree for the $\mathcal{O}(b)$ active components, whose r'-costs and ℓ'-lengths are bounded by those of $\tilde{T}$. By Lemma B.1 the ℓ'-weight of $\tilde{T}$ is no more than OPT_b. The claim for Step 6 now follows from Lemma 5.3. □

Lemma B.4 At each iteration i of the algorithm, the total ℓ'-length of the approximate solution to the DCS problem in Step 4c in this iteration is at most $(1+\varepsilon)^2\mathrm{OPT}_b$.

Proof: Fix an iteration i and let $\mathcal{C}_i = \{C_1, \dots, C_{p_i}\}$ be the active components in this iteration. We show that there exists a solution of ℓ'-cost at most OPT_b to the DCS problem set up in the iteration.

Consider again the Steiner tree $\tilde{T}$ of bounded degree b. Lemma B.1 ensures that its ℓ'-weight is no more than OPT_b, while its r'-cost is bounded from above by $(1+\varepsilon)\mathcal{B}$. Contract the nodes in each of the connected components of (V, F) to a single vertex. It is then straightforward to obtain a Steiner tree T_i from $\tilde{T}$ spanning all the active components and of ℓ'- and r'-cost no more than those of $\tilde{T}$. However, T_i might have nodes of degree larger than b.

Assume now for the sake of simplicity that $|\mathcal{C}_i|$ is even.[8] We apply Lemma 4.6 to T_i where we mark all the active components. This yields a pairing of the active components such that all paths between the nodes are pairwise edge-disjoint. We now construct a feasible solution H to the DCS problem:

(i) First, consider all pairs C_u, C_v of active components such that the path between them in T_i consists of a single edge $e' = (C_u, C_v)$. This edge stems from an edge $(u, v) \in \tilde{T}$, where $u \in C_u$ and $v \in C_v$. By the construction Step 4(b)ii, there is an edge (C_u, C_v) in G_i of ℓ'-length $\ell'(e')$ and r'-cost $r'(e')$. We include this edge (C_u, C_v) in H.

(ii) Now, look at pairs of components C_u, C_v between which the paths in T_i consist of more than a single edge. Each node on such a path is either an active component or a node which is not contained in any active component. Without loss of generality let the path in T_i be such that starting with C_u it first hits only active components and then original nodes before ending in C_v. Formally, let the path be

$$(C_u, C_1), (C_1, C_2), \dots, (C_{q-1}, C_q), (C_q, w_1), (w_1, w_2), \dots, (w_r, C_v)$$

where the C_i are active components and the w_j are original nodes. For each edge (C_i, C_{i+1}) in this path, there is a corresponding edge in $\tilde{T}$ from which this edge is derived (see Figure 6 on the following page for an illustration). Let these edges be $(z_u, b_1), (a_1, b_2), \dots, (a_q, z_v)$. For all j, the vertices b_j and a_j are in the same active component. Thus, they either coincide or there is an edge in G_i of zero ℓ'-length and r'-cost joining them. We define a path P_{C_u,C_v} in G_i as follows:

$$P_{C_u,C_v} := (C_u, b_1), (b_1, a_1), (a_1, b_2), \dots, (a_{q-1}, b_q), (b_q, a_q), (a_q, w_1), (w_1, w_2), \dots, (w_r, C_v)$$

where we include the edge (b_j, a_j) only if $a_j \neq b_j$. This way we find a path P_{C_u,C_v} between every pair C_u, C_v in the pairing where the path in T_i consists of more than one edge. We add those edges to H which occur in an odd number of the paths P_{C_u,C_v}.

Since the original paths were edge-disjoint it follows by the construction above that the total ℓ'-cost of all the edges added to H is no more than OPT_b, while the r'-cost of H does not exceed $(1+\varepsilon)\mathcal{B}$.

We now verify that the degree constraints for the DCS problem are satisfied by H. We must show that all active components C_j have degree equal to one and that the degree of each node $v \in V$ is an even number between 0 and $2b$.

[8] The case when $|\mathcal{C}_i|$ is odd can be treated similarly by adding the dummy node included in Step 4(b)iii to the set $\mathcal{C}_i$.

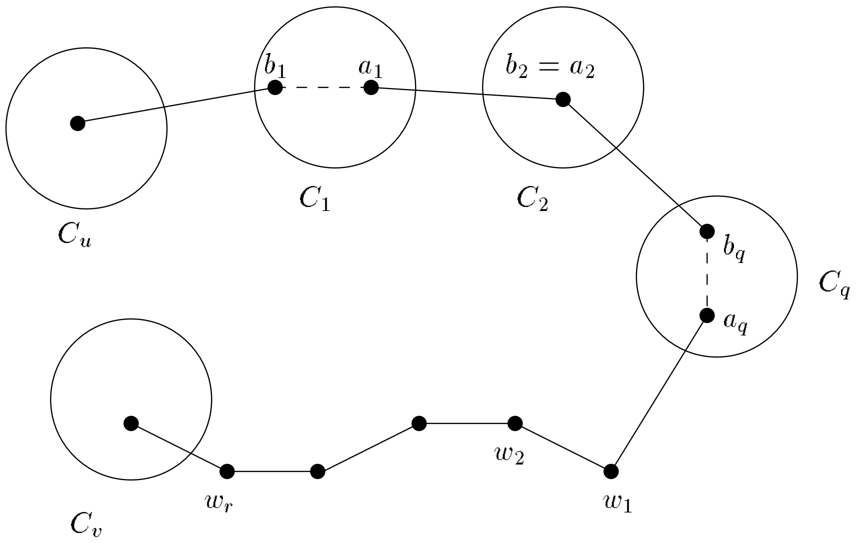

Figure 6: Converting the paths in $\tilde{T}$ to paths in G_i.

The fact that each cluster node C_j has degree equal to one in H follows from the fact that we have found a pairing of the cluster nodes and each node occurs only once and then as an end node in all the paths constructed in the two steps above.

Recall that each vertex $v \in V$ has degree at most b in $\tilde{T}$. The only place where we add edges to H that are adjacent to original vertices from V and which are not in $\tilde{T}$ is in the second step above. By construction, we add at most one extra edge of zero costs for every edge of $\tilde{T}$ incident on v. Thus, there could be at most $2b$ edges incident with v. Since we chose only edges that occurred an odd number of times in the paths, the degree of v is clearly bounded by $2b$. Observe also that all vertices $v \in V$ occur only as internal nodes of the paths that are constructed in the second step. Thus, since we add only edges to H that occur in an odd number of paths, the degree of each such vertex in H must be even.

Thus, we have shown that there is a solution to the DCS problem in Step 4c of total ℓ'-length at most $(1+\varepsilon)\mathrm{OPT}_b$ and of r'-cost at most $(1+\varepsilon)\mathcal{B}$. Thus, the DCS found in Step 4c of Algorithm Improve-Deg-BCST has total r'-cost at most $(1+1/\varepsilon)(1+\varepsilon)\mathcal{B}$ and total ℓ'-length at most $(1+\varepsilon)^2\mathrm{OPT}_b$. □

To complete the analysis, observe that the DCS found in Step 4c of Algorithm Improve-Deg-BCST has r'-cost at most $(1+1/\varepsilon)(1+\varepsilon)\mathcal{B}$. Also, the Steiner tree which is found in Step 7 has r'-cost that is bounded by twice this value. Using these results in conjunction with the result from Lemma B.2 that the number of phases is $\mathcal{O}(\log \frac{|K|}{b})$, we can conclude that the total r'-cost of the edges added to the solution is bounded as specified in Theorem 5.5. Clearly, this is also a bound for the r'-cost of the tree T' computed in Step final-step.

Using Lemmas B.3 and B.4 with the number of iterations, yields the corresponding bound on the ℓ'-length of T'. The degree bound is a consequence of Lemma B.2. By construction of the reduction r and the Steiner tree T in Step 7 the theorem now follows. □