CONF-9501107--1

LA-UR 95-2925

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

TITLE: **NETWORK IMPROVEMENT PROBLEMS**

AUTHOR(S): S.O. Krumke, H. Noltemeier, K.U. Drangmeister, M.V. Marathe, S.S. Ravi

SUBMITTED TO: ACM Symposium on Discrete Algorithms
Atlanta, GA
January, 1995

Los Alamos

Los Alamos National Laboratory
Los Alamos New Mexico 87545

Submitted to SODA 95

Network Improvement Problems

S. O. Krumke[1] H. Noltemeier[1] K. U. Drangmeister[1] · M. V. Marathe[2] S. S. Ravi[3]

July 14, 1995

Abstract

We study *budget constrained optimal network improvement problems.* Such problems aim at finding optimal strategies for improving a network under some cost measure subject to certain budget constraints. As an example, consider the following prototypical problem: Let $G = (V, E)$ be an undirected graph with two cost values $L(e)$ and $C(e)$ associated with each edge e, where $L(e)$ denotes the length of e and $C(e)$ denotes the cost of reducing the length of e by a unit amount. A reduction strategy specifies for each edge e, the amount by which $L(e)$ is to be reduced. For a given budget B, the goal is to find a reduction strategy such that the total cost of reduction is at most B and the minimum cost tree (with respect to some measure $\mathcal{M}$) under the modified L costs is the best over all possible reduction strategies which obey the budget constraint. Typical measures $\mathcal{M}$ for a tree are the total weight and the diameter.

We provide both hardness and approximation results for the two measures $\mathcal{M}$ mentioned above. For the problem of minimizing the total weight of a spanning tree, we provide an algorithm that, for any fixed $\gamma, \varepsilon > 0$, finds a solution whose weight is at most $(1 + \frac{1}{\gamma})$ times that of a minimum length spanning tree plus an additive constant of at most ε and the total cost of improvement is at most $(1 + \gamma)$ times the budget B. This result can be extended to obtain approximation algorithms for more general network design problems considered in [GW, GG+94]. For the problem of obtaining a minimum diameter tree, we present an approximation algorithm that finds a spanning tree with diameter at most $O(\log n)$ times the diameter of a budget constrained optimal diameter spanning tree by spending at most $O(\log n)$ times the budget B, where n denotes the number of nodes in the network.

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

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

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

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

1 Introduction

Given a graph G, the problem of finding a spanning tree minimizing some measure $\mathcal{M}$ is one of the classical problems in computer science. Typical measures $\mathcal{M}$ for the tree T include the total weight $\omega(T)$, the diameter $\mathrm{dia}(T)$, the weighted sum $\omega(x,T)$ of the distances to a fixed node x, etc. However, there has been little work on how to *modify* a graph G such that the lightest spanning tree with respect to the measure $\mathcal{M}$ in the resulting graph has minimum weight. In this paper, we study network design problems where the goal is to find optimal improvement strategies for modifying a given network.

Such problems arise in diverse areas such as design of high speed communication networks [KJ83], video on demand [KPP93], teleconferencing [KPP92, Komp], VLSI design [CK+92, CR91, ZPD94], database retrieval [Ra94], etc. For example, consider the following problem which arises in the design of high speed communication networks. We are given a communication network (modeled as an undirected graph) which needs to be upgraded due to the availability of faster communication links. The cost value $C(e)$ denotes the cost incurred in improving the delay in sending packets using edge e by one unit. The constraint is that the total cost of improving the network cannot exceed a given budget B. The goal of the budget constrained optimal improvement problem is to find a way to upgrade the communication network so that the total upgrading cost is at most B and the cost of a minimum spanning tree in the modified network is minimized.

Most of the network improvement problems considered in this paper are $\mathcal{NP}$-hard. Given these hardness results, we aim at finding efficient approximation algorithms for these problems. Define an (α,β)–approximation algorithm as a polynomial–time algorithm that produces a solution within α times the optimal function value, violating the budget constraint by a factor of at most β. The main contribution of the paper is a framework for formulating such network improvement problems and a collection of efficient (α,β)–approximation algorithms for several versions of network improvement problems. In the next two sections we formally define the problems considered in this paper and summarize our results.

2 Definitions and Preliminaries

Let $G=(V,E)$ be an undirected graph. Associated with each edge $e \in E$, there are three nonnegative values as follows: $L(e)$ 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 and $L_{min}(e)$ denotes the *minimum length* to which the edge e can be reduced. Consequently, we assume throughout the presentation that $L_{min}(e) \leq L(e)$. For each edge $e \in E$, let $\Delta(e)$ be defined by $\Delta(e) := L(e) - L_{min}(e)$.

Given a weighted graph G as above and a budget B, we define a *feasible reduction* to be a nonnegative function r defined on E with the following properties: For all edges $e \in E$, $L(e) - r(e) \geq L_{min}(e)$ and $\sum_{e\in E} r(e)\cdot C(e) \leq B$.

For a feasible reduction r, we denote by $r(G)$ the graph consisting of the same nodes and edges as G with each edge e having length $L_{fin}(e)$ given by $L_{fin}(e) = L(e) - r(e)$.

For some versions of the problems that are discussed in the sequel, we impose some additional constraints on permissible reductions. Thus we obtain the following three cases:

1. For each edge e, the reductions are required to either shorten the length of the edge to $L_{min}(e)$ or not to change the length of edge e at all. Formally, we require each (feasible) reduction to satisfy the condition $r(e) \in \{0, \Delta(e)\}$ for all $e \in E$. These reductions will be referred to as *0/1-reductions*.

2. The reduction r must be an integer valued function; i.e., for each edge e, $r(e)$ must be an integer in $[0, \Delta(e)]$. We denote this type of reductions by *I–reductions* (integer reductions).

3. The third case is the least restricted one. Here we allow a reduction r to take on rational values; i.e., for each edge e, the reduction can be a rational value in $[0, \Delta(e)]$. We refer to these reductions as *C–reductions* (continuous reductions).

An alternative way to view a 0/1–reduction r is to use it to model the *insertion of alternative edges* to the graph G, with the reduction of the edge e corresponding to the construction of a new edge $\hat{e}$ parallel to e with length $L_{min}(e)$.

Let G be a graph and T be a spanning tree of G. The *total length* of T under the weight function L, denoted by $\omega_L(T)$, is defined to be the sum of the lengths of the edges that are in T. The *diameter* of T under the weight function L, denoted by $\mathrm{dia}_L(T_G)$, is the length of a longest simple path in T. We denote the total weight of a minimum total length spanning tree and the diameter of a minimum diameter spanning tree by $MST(G)$ and $MDST(G)$ respectively. We are now ready to state the problems studied in this paper.

Definition 2.1 *The* Budget Minimum Total Cost Spanning Tree Problem (BMST) *is to find a feasible reduction r such that $MST(r(G))$ has the least possible value (under the L_{fin} function).*

Definition 2.2 *The* Budget Minimum Diameter Spanning Tree Problem (BMDST) *is to find a feasible reduction r such that $MDST(r(G))$ has the least possible value (under the L_{fin} function).*

Let $f : \mathbb{N} \to \mathbb{Q}$ be a function. We say that an algorithm A has a *relative performance guarantee* of $f(n)$, if for each instance given by an n node graph G, the reduction r provided by A satisfies

$$\frac{MST(r(G))}{MST(r^*(G))} \leq f(n). \tag{1}$$

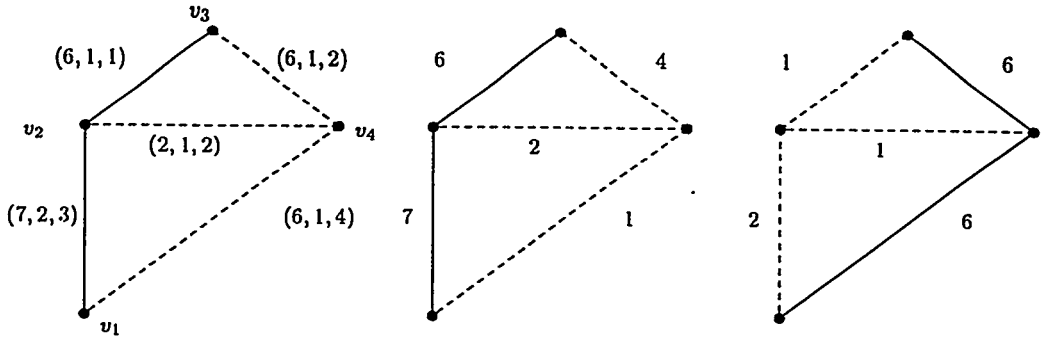
Here r^* denotes an optimal edge–reduction on G and n denotes the number of nodes in G.

For the remainder of the paper, we use the following convention: Given a problem $\mathcal{P}$, let 0/1–$\mathcal{P}$, I–$\mathcal{P}$ and C–$\mathcal{P}$ denote the problems when the reduction r is restricted to be a 0/1-reduction, I-reduction and C-reduction respectively.

Example: Consider the graphs given in Figure 1. Figure 1(a) shows a graph G where each edge is associated with the three values (L, L_{min}, C). The result of a modification of G is shown in Figure 1(b). The edges belonging to the minimum spanning tree are drawn as dashed lines. The modification corresponding to Figure 1(b) involves a cost of 24 and the weight of the resulting tree is 7. Figure 1(c) shows the graph with edge lengths resulting from a reduction that is optimal among all reductions of cost no more than 22. There, the weight of the spanning tree resulting from the reduction is 4. Thus, the reduction of Figure 1(b) is a $(7/4, 24/22)$–approximation to an optimal solution.

3 Summary of Results

The results obtained in this paper are summarized in the following tables.



(a) The original graph G

(b) Modification of G with cost 24

(c) Optimal Modification for a budget of 22

Figure 1: An example of a graph modification via edge reductions.

BMST Problem			
	0/1	I	C
Trees	weakly $\mathcal{NP}$–hard FPAS	easy (Greedy)	easy (Greedy)
General Graphs	$\mathcal{NP}$–hard $(1+1/\gamma, 1+\gamma)$–approx.[4]	$\mathcal{NP}$–hard $(1+1/\gamma, 1+\gamma)$–approx.[4]	$\mathcal{NP}$–hard $(1+1/\gamma, 1+\gamma)$–approx.[4]

BMDST Problem			
	0/1	I	C
Trees	weakly $\mathcal{NP}$–hard	open	open
General Graphs	strongly $\mathcal{NP}$–hard	strongly $\mathcal{NP}$–hard	open
	also hard to approx. within $(11/10-\varepsilon, (1/8-\varepsilon')\log n)$	also hard to approx. within $(11/10-\varepsilon, (1/8-\varepsilon')\log n)$	open
	$(O(\log n), O(\log n))$–approx.	$(O(\log n), O(\log n))$–approx.	$(O(\log n), O(\log n))$–approx.

The approximation algorithm for C–BMST can be extended significantly. For example, using our ideas in conjunction with the results of Goemans et. al. [GG+94], we can obtain similar approximation results for finding budget constrained minimum-cost generalized Steiner trees, minimum-cost k-edge connected subgraphs and other network design problems specified by weakly supermodular functions introduced in that paper.

The remainder of the paper is organized as follows. In Section 4 we briefly discuss the structure of an optimal solution. Section 5 contains the hardness results for the various problems considered in the paper. Sections 6 and 7 contain our approximation algorithms.

[4]The length of the spanning tree produced is at most $(1+1/\gamma)$ times that of an optimal tree plus an additive constant of ϵ that can be made arbitrarily close to zero.

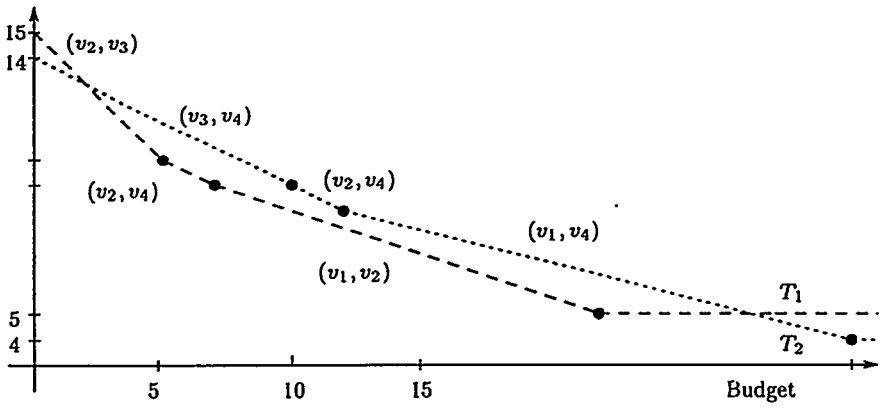


Figure 2: Remaining weight of the trees T_1 and T_2 as a function of the budget.

4 Structure of an Optimal Solution

In this section we comment on the structure of optimal solutions to both BMST– and BMDST–problems. We also look at special cases of these problems that can be solved in polynomial time.

First, suppose that the given budget B is zero. Then BMST reduces to the well known minimum spanning tree problem (with length function $L(e)$), and is known to be optimally solvable by classical algorithms, e.g. Prim's algorithm [CLR]. Also, BMDST can be solved in polynomial time, e.g. by solving the so–called "Absolute 1–Center Problem" (cf. [Ha95]).

Similarly, if $B = +\infty$ (i.e., there is no bound on the cost of upgrading the network), the BMST and BMDST problems again reduce to the minimum spanning tree problem and the minimum diameter spanning tree problem respectively (with length L_{min}).

Optimal solutions to BMST and BMDST problems also exhibit some structure in the general case (i.e., $B \notin \{0, +\infty\}$). Due to lack of space, we restrict our discussion to the BMST–problem. A similar argument can be given for the BMDST–problem.

Any (feasible) reduction r induces a tree in the natural way, namely a minimum spanning tree T_r in the modified graph $r(G)$. Observe that the quality of the solution produced via the reduction r depends solely on the weight of T_r, so all the cost incurred in upgrading edges not in T_r is wasted. Moreover, for any *fixed* tree T in G, the Greedy–strategy that successively reduces a cheapest available edge is an optimal (continuous or integer) reduction strategy. Thus, if we already knew a minimum spanning tree T_{r^*} corresponding to an optimal reduction r^*, we could solve C–BMST and I–BMST in polynomial time.

This observation also suggests a very simple exponential time algorithm for solving the two versions of the BMST problem mentioned above: Enumerate all spanning trees in G, apply the Greedy strategy to each of them and then select the best solution. Unfortunately, a graph G with n nodes can have n^{n-2} different spanning trees.

We now discuss the sensitivity of optimal (continuous) reduction strategies to changes in the given budget B. If we fix a spanning tree and plot the weight of that tree as a function of the money spent on it in a Greedy manner, we see that this function is piecewise linear and convex. Each piece corresponds to a budget range, where one particular edge e is shortened. Thus it is easy to see that the piece has slope $-1/C(e)$.

Figure 2 shows the plots corresponding to the tree T_1 consisting of the edges (v_2, v_3), (v_2, v_4), (v_1, v_2) and the tree T_2 consisting of the edges (v_3, v_4), (v_2, v_4) and (v_1, v_4) taken from the example graph of Figure 1. As can be seen from Figure 2, the plots for different trees can cross each other multiple times. If we plot the weights of all spanning trees on the same set of axes, the lower envelope gives the optimal remaining weight per budget. It is easy to see that the lower envelope can have an exponential number of linear pieces.

As far as we know, the problems considered in this paper have not been previously studied. Berman [Be92] considers the problem of shortening edges in a given tree to minimize its shortest path tree weight. Phillips [Ph93] 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 [Ph93, Be92] can be broadly classified as types of bicriteria problems. Recently, there has been substantial work on finding efficient approximation algorithms for a variety of bicriteria problems (see [KP95, Ha92, MRS+95, RR+93, Ra94, Wa92, ZPD94] and the references therein).

5 Hardness Results

5.1 Hardness of the BMST–Problem

It is easy to see that when G is a tree, C–BMST and I–BMST problems can be solved optimally in polynomial time by a Greedy–type algorithm that simply keeps on reducing the length of the cheapest available edge. In contrast, 0/1–BMST is hard even for trees as indicated in the following proposition which can be proven using a reduction from the KNAPSACK problem [GJ79].

Proposition 5.1 *0/1–*BMST *is $\mathcal{NP}$–hard, even if the underlying network G is a tree. This result remains true, even if $C(e) = 1$ and $L_{min}(e) = 1$ for all $e \in E$.* □

We now turn to the complexity of BMST for *general graphs*. Again using a reduction from KNAPSACK, we can prove the following.

Proposition 5.2 *For general graphs, I–*BMST *is $\mathcal{NP}$–hard, even if $C(e) = 1$ for all $e \in E$.* □

While it might be not surprising that the discrete versions of BMST turn out to be hard, it is also the case that the continuous version, C–BMST, cannot be solved optimally in polynomial time, even for very restricted classes of graphs, unless $\mathcal{P} = \mathcal{NP}$.

Proposition 5.3 *C–*BMST *is $\mathcal{NP}$–hard, even when restricted to series–parallel graphs.*

Proof: We use a reduction from CMC–KNAPSACK[5] (c.f. [GJ79] p. 247). An instance of CMC–KNAPSACK is given by a finite set U of n items, a size $s(u)$ and value $v(u)$ for each item, a partition $U_1 \cup \cdots \cup U_k$ of U into disjoint sets and two integers S and K. The question is, whether there is a choice of a unique element $u_i \in U_i$, for each $1 \leq i \leq k$, and an assignment of rational numbers $r_i, 0 \leq r_i \leq 1$ to these elements such that $\sum_{i=1}^{k} r_i s(u_i) \leq S$ and $\sum_{i=1}^{k} r_i v(u_i) \geq K$.

Given an instance of CMC–KNAPSACK we construct a graph $G = (V, E)$ in the following way: We let $V = U \cup \{X, T, T_1, \ldots, T_k\}$, $E := E_1 \cup E_2 \cup E_3$ with $E_1 := \{(X, u) : u \in U\}$, $E_2 := \{(u, T_i) : u \in U_i, i =$

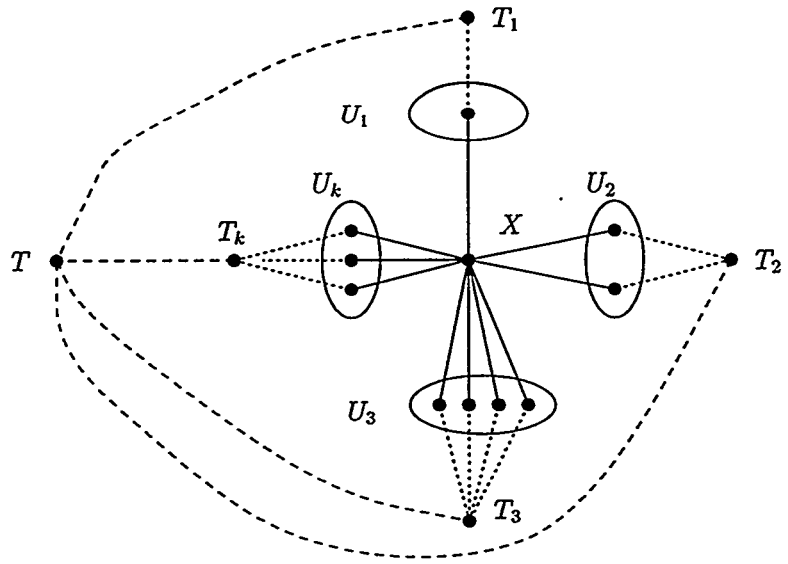


Figure 3: Graph used in the reduction from CONTINUOUS MULTIPLE CHOICE KNAPSACK.

$1,\ldots,k\}$ and $E_3 := \{(T_i,T) : i = 1,\ldots,k\}$. The graph constructed this way is obviously series–parallel with the terminals X and T.

Define $D := \max\{v(u) : u \in U\}$. For each edge $(x,u) \in E_1$, let $L(x,u) := D, L_{min}(x,u) := D - v(u), C(x,u) := s(u)/v(u)$. For all edges $e \in E_2$ we let $L(e) := L_{min}(e) := C(e) := 0$, and for all edges $e \in E_3$ we define $L(e) := L_{min}(e) := 3D$ and $C(e) := 0$. Set the bound B on the total cost to be S.

The graph is shown in Figure 3. The dotted edges are of weight 0 while the dashed ones have weight $3D$. Any MST in G has weight $kD + 3D$.

By the construction, any feasible reduction can only reduce the length of the edges in E_1. Assume that r is a feasible reduction. Observe that the MST in $r(G)$ will always include *all* edges from E_2 (which are of weight 0) and *exactly one* edge from E_3, regardless of which edges from E_1 are affected by the reduction. Observe also that for any fixed $i \in \{1,\ldots,k\}$ the MST in $r(G)$ will contain *exactly one* of the edges of the form (X,u') $(u' \in U_i)$. Consequently, reducing the length of *more than one* edge (X,u') and $u' \in U_i$ will *not* improve the quality of the solution, but cost money from the budget B. We thus have:
Observation: *If r is a feasible reduction for the instance of* BMST *defined above and the weight of an MST in $r(G)$ is C, then there is always a feasible reduction r', which for each $i \in \{1,\ldots,k\}$ reduces at most one of the edges (X,u), $u \in U_i$ and the weight of an MST in $r'(G)$ is also equal to C.*

Let r be any reduction as defined in the above observation and for $i = 1,\ldots,k$ let $e_i = (X,u_i)$ be the unique edge from x to U_i affected by the reduction. The weight of an MST in $r'(G)$ is then given by

$$3D + \sum_{i=1}^{k}(L(e_i) - r(e_i)) = 3D + k \cdot D - \sum_{i=1}^{k} r(e_i). \qquad (2)$$

The cost of reduction r is given by

$$\sum_{i=1}^{k} r(e_i)C(e_i) = \sum_{i=1}^{k} r(e_i) \cdot \frac{s(u_i)}{v(u_i)} = \sum_{i=1}^{k} \frac{r(e_i)}{v(u_i)} \cdot s(u_i) \leq B. \qquad (3)$$

[5] CONTINUOUS MULTIPLE CHOICE KNAPSACK

We now prove the following: There is a feasible reduction r such that an MST in $r(G)$ has weight at most $(3+k)D-K$, if and only if there exists a choice of a unique element $u_i \in U_i$, $1 \leq i \leq k$ and an assignment of rational numbers $r_i, 0 \leq r_i \leq 1$ to these elements such that $\sum_{i=1}^{k} r_i s(u_i) \leq B$ and $\sum_{i=1}^{k} r_i v(u_i) \geq K$.

First, assume that there is a feasible reduction r such that the MST in $r(G)$ has weight at most $(3+k)D-K$. Without loss of generality, we can assume that r has the properties as stated in the above observation. Then for $i = 1, \ldots, k$ there is at most one edge $e_i = (X, u)$ with $u \in U_i$ such that $r(e_i) > 0$. If there is such an edge e_i, we define

$$r_i := \frac{r(e_i)}{v(u_i)} = \frac{r(e_i)}{L(e_i) - L_{min}(e_i)} \tag{4}$$

and let $u_i := u$. If for all edges (X, u) with $u \in U_i$ we have $r(e_i) = 0$, we simply let $r_i := 0$ and choose $u_i \in U$ arbitrarily. It follows readily from the definition and the feasibility of the reduction r that $r_i \in [0, 1]$. Moreover, using Equation (3) we see that $\sum_{i=1}^{n} r_i s(u_i) \leq B \leq B_1$. Using Equation (2) and the fact that the weight of the MST in $r(G)$ is no more than $(3+k)D-K$ we obtain that

$$\sum_{i=1}^{n} r_i v(u_i) = \sum_{i=1}^{n} \frac{r(e_i)}{v(u_i)} \cdot v(u_i) = \sum_{i=1}^{n} r(e_i) \geq K.$$

Conversely, if we can pick unique elements u_i from the sets U_i and find rational numbers $r_i \in [0, 1]$ such that $\sum_{i=1}^{n} r_i s(u_i) \leq B$ and $\sum_{i=1}^{n} r_i v(u_i) \geq K$. We can define a reduction r by $r(X, u_i) := r_i v(u_i) = r_i(L(x, u_i) - L_{min}(x, u_i))$ for $i = 1, \ldots, k$ and $r(e) := 0$ for all other edges. It follows that r is indeed feasible, and using Equation (2) we see that the MST in $r(G)$ is no heavier than $(3+k)D-K$. □

5.2 Hardness of the BMDST–Problem

As in the case of the BMST–problem, we obtain the following:

Proposition 5.4 *0/1-*BMDST *is $\mathcal{NP}$-hard for trees and even if $C(e) = 1$ for all $e \in E$.* □

The proof is again by a reduction from KNAPSACK and is omitted in this extended abstract. The following proposition presents a negative result concerning the approximability of I–BMDST and 0/1–BMDST. A proof of this proposition is included in the appendix.

Proposition 5.5 *Let $\varepsilon > 0$ be any positive constant. If there is a polynomial time approximation algorithm A for I-*BMDST *(0/1-*BMDST*) with a performance guarantee of $(11/10 - \varepsilon)$, then $\mathcal{P} = \mathcal{NP}$. This results remains true even if $C(e) = 1$ for all $e \in E$. Moreover, unless $\mathcal{NP} \subseteq DTIME(n^{\log \log n})$, the above hardness results hold even if the algorithm A is allowed to spend $\mu B \log_2 B$ units of money instead of B for any fixed μ, $0 < \mu < 1/8$.* □

6 Approximation Algorithms for BMST–Problems

In this section, we present our approximation algorithm for the BMST problem. As mentioned earlier, the approximation algorithm extends easily to a broad class of network improvement problems where the objective to be minimized is the total cost of a connected subnetwork (e.g. the case of budget constrained

Procedure Heuristic–BMST(γ, ε)

1 Use binary search to find the smallest integer $j' \in [0, \frac{(n-1)(\max_{e\in E} L(e) - \min_{e\in E} L_{min}(e))}{\gamma\varepsilon}]$ such that for $C' := \frac{(n-1)\min_{e\in E} L_{min}(e)}{\gamma} + j'\varepsilon$ the procedure Test(C') returns Yes.

2 Let T' be the tree generated by Test(C').

3 Define the reduction r by $r(e) := 0$ if e is not included in T and by $r(e) := t_e$ otherwise.

4 **return** r and T.

Figure 4: Main Procedure for the approximation of BMST

Procedure Test(C)

1 Let $\mu := \frac{C}{B}$

2 **for** each edge e let $h(e) = \min_{t\in[0,\Delta(e)]} (L(e) + t(\mu C(e) - 1))$.
Also, let t_e be the value of t which achieves the value $h(e)$. (See Section 6.1.1)

3 Compute a minimum spanning tree T in G using the weight $h(e)$ for each $e \in E$.

4 **if** $\omega_h(T) \leq (1+\gamma)C$ **then return** Yes **else return** No.

Figure 5: Test procedure used for the approximation of BMST

minimum Steiner tree problem). The approximation algorithm uses a parametric search technique similar to the one devised in [MRS+95].

Our approximation algorithm for 0/1–BMST is shown in Figure 4. This algorithm uses the test procedure given in Figure 5.

6.1 Correctness and Performance Guarantee

The performance guarantee provided by the algorithm Heuristic–BMST is summarized in the following theorem.

Theorem 6.1 *For any fixed $\gamma, \varepsilon > 0$, Heuristic–*BMST *is a polynomial time approximation algorithm for 0/1–*BMST *(I–*BMST *and C–*BMST *respectively) that finds a solution whose length is at most $(1 + \frac{1}{\gamma})$ times that of a minimum length spanning tree plus an additive constant of at most ε, and the total cost of improvement is at most $(1+\gamma)$ times the budget B.*

The proof of Theorem 6.1 relies on several lemmas presented below.

Imagine the (possibly infinite) graph $\tilde{G} = (V, \tilde{E})$ that has the same node set V as G but for each edge $e \in E$ has a number of parallel edges e^t, one for each possible value $t \in [0, \Delta(e)]$ to which the length $L(e)$ can be reduced. In the case of integer–reductions, we restrict t to the integers in the range $[0, \Delta(e)]$; for 0/1–reductions, we restrict t to the two element set $\{0, \Delta(e)\}$.

To be precise, let $\tilde{E} = \bigcup_{e\in E} E_e$, where $E_e := \{e^t : 0 \leq t \leq \Delta(e)\}$. Define $length(e^t) := L(e) - t$ and $cost(e^t) := tC(e)$. Now we can view the problem of finding an optimal reduction for our original graph as finding a spanning tree T in $\tilde{G}$ that has $cost(T) \leq B$ and has minimum total length $length(T)$ among all spanning trees obeying the cost–constraint. We will base our proofs on $\tilde{G}$ rather than on G to make the argument a little bit easier.

For a fixed value of μ denote by $T_h := MST_h(\tilde{G})$ the minimum spanning tree of $\tilde{G}$ with respect to the single weight function $h(e^t) := length(e^t) + \mu \cdot cost(e^t) = (L(e) + t(\mu C(e) - 1))$. Let $t_e \in \{0, \Delta(e)\}$ be

chosen so that $(L(e) + t(\mu C(e) - 1))$ is minimized. Observe that any spanning tree can include at most one of the edges from each set E_e. Using these facts, the following lemmas can be proven.

Lemma 6.2 *If T_h includes an edge from E_e, this edge must be e^{t_e}.* □

Lemma 6.3 *The function $\mathcal{R}(C) = \frac{\omega_h(MST(\tilde{G}))}{C}$ is monotonically nonincreasing on $\mathbb{Q} \setminus \{0\}$.* □

Proof of Theorem 6.1: Let r^* be an optimal feasible reduction and let T^* be a minimum spanning tree in $r^*(G)$ of weight $\omega(T^*)$. Consider the call to the procedure Test when C is some rational number C^* satisfying $C^* = \omega(T^*)/\gamma + \varepsilon'$ with $0 \leq \varepsilon' \leq \varepsilon$. Observe that by Step 1 of the algorithm, which searches the interval $[\frac{1}{\gamma}(n-1)\min_{e\in E} L_{min}(e), \frac{1}{\gamma}(n-1)\max_{e\in E} L(e)]$ with a spacing of ε, such a call will be made.

For each edge $e \in T^*$ we can identify the corresponding edge $e^{r^*(e)}$ in $\tilde{G}$. This way, we obtain a tree T in $\tilde{G}$ of h–weight no more than $\omega(T^*) + \mu B = \omega(T^*) + \frac{C^*}{B}B$. Consequently, the minimum spanning tree with respect to h that is found by the procedure during this call has h–weight at most $\omega(T^*) + C^* = \gamma(\frac{\omega(T^*)}{\gamma} + \varepsilon') - \gamma\varepsilon' + (\frac{\omega(T^*)}{\gamma} + \varepsilon') \leq (1+\gamma)\left(\frac{\omega(T^*)}{\gamma} + \varepsilon'\right) = (1+\gamma)C^*$.

Thus, we observe that the procedure will return Yes and that $\mathcal{R}(C^*) \leq 1+\gamma$. Further, the value C' found by the algorithm satisfies $C' \leq C^*$, since C' is the minimum value such that Test(C') returns Yes.

Let $T_{C'}$ be the minimum spanning tree found by Test when C equals C'. Then we have

$$\omega_{L_{fin}}(T_{C'}) \leq \omega_h(T_{C'}) \leq \omega(T^*) + \frac{C'}{B}B \leq \omega(T^*) + C^* = \omega(T^*) + \frac{\omega(T^*)}{\gamma} + \varepsilon' \leq (1 + \frac{1}{\gamma}) \cdot \omega(T^*) + \varepsilon.$$

Moreover, $\frac{C'}{B}cost(T_{C'}) \leq \omega_h(T_{C'}) \leq C'(1+\gamma)$; that is, $cost(T_{C'}) \leq (1+\gamma)B$. □

6.1.1 Running Time

We now show that the algorithm can be implemented to run in polynomial time. For this, it suffices to show that each execution of procedure Test(C) can be completed in polynomial time. Consider the execution of the procedure Test for a given value of μ. Observe that in Step 2, we choose a value of $t = t_e$ to minimize $f_e(t) := (L(e) + t(\mu C(e) - 1))$. By the linearity of f_e it follows that the value of t that minimizes f_e is either 0 or $\Delta(e)$. As an immediate consequence, we get that Step 2 of Procedure Test can be carried out in constant time. Let the graph have m edges and n vertices. Then the total running time of Procedure Test is $O(n + m\log\beta(m,n))$ (using the algorithm of Gabow et. al. [GGS86] for finding a minimum spanning tree), where $\beta(m,n) = \min\{i \mid \log^{(i)} n \leq m/n\}$. Let $L_{max} = \max_{e\in E} L(e)$. Since the total number of calls to Procedure Test is bounded by $O(\log(\frac{L_{max}}{\gamma\varepsilon}))$, the total running time of the algorithm is $O(\log(\frac{L_{max}}{\gamma\varepsilon})(n + m\log\beta(mn)))$. Since γ and ε are fixed, the running time is polynomial in the size of the problem instance.

7 Approximation Algorithms for BMDST–Problems

Next, we discuss our approximation algorithms for BMDST–problems. For this we first recall the following theorem from [MRS+95].

Theorem 7.1 *There is a polynomial-time algorithm that, given an undirected graph G on n nodes with nonnegative integral cost d and c on its edges, a bound D, and a fixed $\varepsilon > 0$, constructs a spanning tree of G of diameter at most $2\lceil\log_2 n\rceil D$ under the d-costs and of total c–cost at most $(1+\varepsilon)\lceil\log_2 n\rceil$ times that of the minimum-c-cost of any spanning tree with diameter at most D under d.* □

We can obtain an approximation algorithm for 0/1–BMDST using Theorem 7.1. We simply double[6] each edge e; the first copy has length $L(e)$ and cost 0, while the second one has length $L_{min}(e)$ and cost $C(e)$. Then we perform a binary search and locate the minimum $D \in [(n-1)\min_{e\in E} L_{min}(e), (n-1)\max_{e\in E} L_{min}(e)]$ such that the algorithm referred to in Theorem 7.1 returns a tree of cost at most $(1+\epsilon)\lceil \log_2 n \rceil B$. By inspecting which copy of an edge e is included in the tree, we decide whether or not we should reduce the length of e. Using this approach, we can approximate the optimum diameter within a factor of $2\lceil \log_2 n \rceil$.

Using scaling techniques, we can extend the above idea to obtain an $(O(\log n), O(\log n))$ approximation algorithm for the I–BMDST and the C–BMDST problems. Due to lack of space, we just sketch the main ideas behind the algorithm.

For each edge e in the graph let b_e be chosen so that $2^{b_e} \leq L(e) - L_{min}(e) \leq 2^{b_e+1}$. We now add parallel edges e^k for $k = 0, \ldots, b_e$, where edge e^k has length $L(e) - 2^k$ and cost $2^k C(e)$. We add one more parallel edge e^{-1} of length $L(e)$ and cost 0. Let G be the original graph and G' be the graph obtained as a result of the transformation. Observe that G' can be computed in polynomial time. Let $B' := 2B$.

We again perform a binary search over the range of possible diameter values to obtain the smallest diameter D such that the algorithm referred to in Theorem 7.1 finds a tree of cost at most $(1+\epsilon)\lceil \log_2 n \rceil B'$. As before, we obtain the reduction by examining which of the parallel edges e^k is included in T.

We now argue that this algorithm will result in an $(O(\log n), O(\log n))$–approximation for both I–BMDST and C–BMDST problems. Let r^* denote the optimal reduction involving a cost of at most B, T^* is a minimum diameter spanning tree in $r^*(G)$. Recall that $\delta^* = \mathrm{dia}(T^*_{r(G^*)})$ denotes the optimal L–diameter of T^*. Also, $\delta_{G'} = \mathrm{dia}(T^*_{G'})$ denotes minimum diameter of a spanning tree in G' for the budget B'.

Let us first understand the relationship between B' and B and that between δ^* and $\delta_{G'}$. Consider the tree T^* in $r^*(G)$. We can define a tree T' in G' in the following way: For an edge $e \in T^*$ that is reduced by $r^*(e)$ we select an edge in G' of length $L(e) - b(e)$, where $b(e)$ is selected in such a way that $b(e)/2 \leq r^*(e) \leq b(e)$. Observe that all the edges selected in the above fashion have their lengths reduced by at most $L(e) - 2 \cdot r^*(e)$. Using this fact, the following claim can be proven.

Claim 7.2 *The cost of tree T' is at most $2B$. The diameter of tree T' in G' is at most δ^*.* □

Hence we have demonstrated a witness tree T' such that if the bound on the cost of this tree is $B' := 2B$ then we will have a diameter of at most δ^*. Consequently an *optimum* diameter tree $\hat{T}$ in G' under the bound B' on the cost will have diameter at most δ^*.

Since the diameter of a graph is a non-increasing function of the total budget on the cost of improvement, it follows that performing binary search yields the desired solution. Specifically, for our algorithm sketched above, the diameter δ of the MDST in $r(G)$, where r is the reduction returned by the heuristic, is bounded from above by $\log_2 n \cdot \mathrm{dia}(\hat{T}) \leq \log_2 n \cdot \delta^*$. As seen before, the total cost of such a solution is no more than $\log n \cdot B' \leq 2 \log n \cdot B$. This shows that the algorithm indeed yields an $(O(\log n), O(\log n))$–approximation for the I–BMDST problem.

It can be argued that the same algorithm also provides a performance of $(O(\log n), O(\log n))$ for the C–MDST problem, i.e., when the reduction is a rational function. Thus, we have the following theorem.

[6] Although the algorithm referred to in Theorem 7.1 does not work for multi-graphs, it can be easily modified to handle multi-edges.

Theorem 7.3 *There is a polynomial time algorithm that, given any instance of 0/1–*BMDST, *I–*BMDST *or C–*BMDST *and for any positive $\epsilon > 0$, finds a reduction r involving cost at most $2(1+\epsilon)\lceil \log_2 n \rceil B$ that yields a minimum diameter spanning tree in $r(G)$ of diameter at most $2\lceil \log_2 n \rceil \delta^*$, where $\delta^* = \mathrm{dia}(T^*)$ denotes the diameter of a minimum diameter spanning tree T^* in $r^*(G)$ and r^* denotes an optimal feasible reduction.* □

References

[Be92] O. Berman, "Improving The Location of Minisum Facilities Through Network Modification," *Annals of Operations Research*, 40(1992), pp. 1–16.

[CK+92] 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.

[CR91] J. P. Cohoon and L. J. Randall, "Critical Net Routing," *IEEE International Conference on Computer Design*, 1991, pp. 174-177.

[CLR] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill Book Co., Cambridge, MA, 1990.

[Cu85] W. Cunningham, "Optimal Attack and Reinforcement of a Network," *J. ACM*, 32(3), 1985, pp. 549-561.

[GGS86] 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.

[GJ79] 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.

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

[GG+94] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, E. Tardos and D. P. Williamson, "Improved Approximation Algorithms for Network Design Problems," *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'94), Jan. 1994, pp. 223-232.

[Ha92] R. Hassin, "Approximation Schemes for the Restricted Shortest Path Problem," *Math. of OR*, 17(1), 1992, pp. 36-42.

[Ha95] R. Hassin and A. Tamir, "On the Minimum Diameter Spanning Tree Problem," *Information Processing Letters*, 53 (2), Jan. 1995.

[KP95] 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.

[KJ83] B. Kadaba and J. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Trans. on Communication*, Vol. COM-31, Mar. 1983, pp. 343-351.

[KPP92] V. P. Kompella, J. C. Pasquale and G. C. Polyzos, "Multicasting for Multimedia Applications," *Proc. of IEEE INFOCOM '92*, May 1992.

[Komp] 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.

[KPP93] V. P. Kompella, J. C. Pasquale and G. C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Transactions on Networking*, 1993, pp. 286–292.

[LY93] 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.

[MRS+95] M. V. Marathe, R. Ravi, S. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III, "Bicriteria Network Design Problems," To appear in *Proc. International Conference on Automata, Languages and Programming* (ICALP'95), July 1995.

[Ph93] C. Phillips, "The Network Inhibition Problem," *Proc. 25th Annual ACM Symp. on Theory of Computing* (STOC'93), May 1993, pp. 288–293.

[RR+93] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz and H. B. Hunt III, "Many Birds with one Stone: Multi–objective Approximation Algorithms," *Proc. 25th Annual ACM Symposium on the Theory of Computing* (STOC'93), May 1993, pp. 438–447. (An expanded version appears as Brown University Technical Report TR-CS-92-58.)

[Ra94] R. Ravi, "Rapid Rumor Ramification: Approximating the Minimum Broadcast Time," *Proceedings of the 35th Annual Symp. Foundations of Computer Science* (FOCS'94), Nov. 1994, pp. 202–213.

[Wa92] A. Warburton, "Approximation of Pareto optima in Multiple–Objective, Shortest Path Problems," *Oper. Res.*, Vol. 35, 1987, pp. 70–79.

[ZPD94] Q. Zhu, M. Parsa and W. Dai, "An Iterative Approach for Delay Bounded Minimum Steiner Tree Construction," Technical Report UCSC-CRL-94-39, Board of Computer Engineering, University of California, Santa Cruz, Oct 1994.

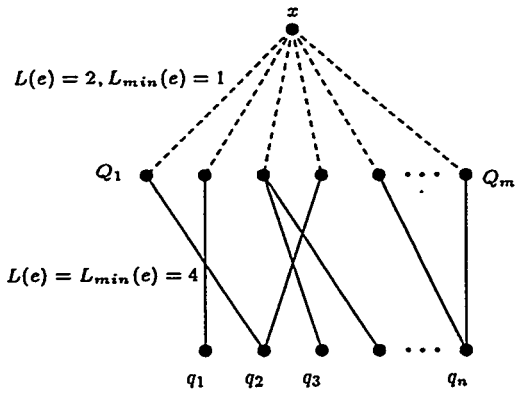


Figure 6: Graph used in the reduction from SET COVER.

8 Appendix

Proof of Proposition 5.1

An instance of KNAPSACK is given by n items $\{a_1, \ldots, a_n\}$ of weight $s(a_i)$, value $u(a_i)$ $(i = 1, \ldots, n)$ respectively and two integers S and U. The question is, whether one can pick items of weight at most S obtaining a value of at least U, i.e. whether there is a subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} s(a_i) \leq S$ and $\sum_{i \in I} u(a_i) \geq U$. It is well known (cf. [GJ79]) that KNAPSACK remains $\mathcal{NP}$–complete, even if $s(a_i) = u(a_i)$ for all items a_i.

Given an instance of KNAPSACK with $s(a_i)u(a_i)$ we construct a star–shaped graph $G = (V, E)$ in the following way: We let $V := \{x, a_1, \ldots, a_n\}$, $E := \{e_1, \ldots, e_n\}$ with $e_i := (x, a_i)$ and define $L(e_i) := 1 + s(a_i), L_{min}(e_i) := 1$ and $C(e_i) := 1, B := S$.

The weight of the MST in G is now $\Omega := n + \sum_{i=1}^{n} s(a_i)$. It is now easy to see that there is a feasible reduction r such that the MST in $r(G)$ has weight at most $\Omega - U$ if and only if there is a subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} s(a_i) \leq S$ and $\sum_{i \in I} s(a_i) \geq U$. □

Proof of Proposition 5.5

We use a reduction from SET COVER (cf. [GJ79], SP5). An instance of SET COVER consists of a set Q of ground elements $\{q_1, \ldots, q_n\}$, a collection $Q_1, \ldots, Q_m$ of subsets of Q and an integer k. The question is whether one can pick at most k sets such that their union equals Q.

Given an instance of SET COVER, we first construct the natural bipartite graph, one side of the partition for set nodes Q_j, $j = 1, \ldots, m$, and the other for element nodes q_i, $i = 1, \ldots, n$. We insert an edge $\{Q_j, q_i\}$ iff $q_i \in Q_j$. All these edges e have length $L(e) = L_{min}(e) = 4$. Now we add a node x and join it with all the set nodes. For these edges e we define $L(e) = 2$, $L_{min}(e) = 1$. Finally, we let $C \equiv 1$ and choose $B = k$. The resulting graph G' is shown in Figure 6.

What we have constructed above yields both an instance of I–BMDST and 0/1–BMDST. Without loss of generality we can assume that there is no single set Q_j covering all the elements in Q, i.e. $Q_j \neq Q$ for $j = 1, \ldots, m$. Then in this case the spanning tree T^* in G' with minimum diameter satisfies $\mathrm{dia}(T^*) = 12$, and a diametric path of that tree is given between any two element nodes that are not adjacent to the same set node.

Observe that any feasible reduction r corresponds to a choice of at most $B = k$ sets from the collection $Q_1, \ldots, Q_m$.

Given any integer reduction r it is easy to see that there is a spanning tree in $r(G')$ with diameter 10, if the selection of sets corresponding to the reduction covers all the elements in Q, and that the diameter of $r(G')$ is at least 11, if the selection does not form a cover.

Consequently, if an algorithm A has a performance guarantee of $(11/10 - \varepsilon)$ and there is a set cover of size k or less then A must return a reduction $A(r)$ yielding a MDST of diameter at most 10. On the other hand, if there is no set cover, then the best tree we can obtain by modify the network has diameter 11. Thus an algorithm A with the properties stated in the first part of the proposition can be used to decide an arbitrary instance of SET COVER.

We now consider the optimization version of SET COVER, which is called MIN SET COVER. It is shown in [LY93] that MIN SET COVER can not be approximated in polynomial time within a factor of $\mu \log_2 n$ for any fixed $0 < \mu < 1/8$ unless $\mathcal{NP} \subseteq DTIME(n^{\log \log n})$.

Given any instance I of MIN SET COVER, we construct the graph G' as above. Then we run the algorithm A for the budgets $B = 1, \ldots, \min\{n, m\}$. Observe that this will still result in an overall polynomial time. Let $B_{\min}$ denote the minimum budget in $\{1, \ldots, \min\{n, m\}\}$ such that A returns a reduction $A(r)$ resulting in a MDST of diameter 10. By the observations from above and the fact that the algorithm spends at most $\mu B \log_2 B$ units of money, we see that there must be a set cover of size at most $\mu B \log_2 B$. By the choice of $B_{\min}$ there can be no set cover of size strictly less than $B_{\min}$. Thus we can approximate the minimum set cover by a factor of no more than $\mu \log_2 B_{\min} \leq \mu \log_2 n$. □