Approximation Algorithms for Network Design with Metric Costs

Joseph Cheriyan* and Adrian Vetta†

Abstract. We study undirected networks with edge costs that satisfy the triangle inequality. Let n denote the number of nodes. We present an $O(1)$-approximation algorithm for a generalization of the metric-cost subset k-node-connectivity problem. Our approximation guarantee is proved via lower bounds that apply to the simple edge-connectivity version of the problem, where the requirements are for edge-disjoint paths rather than for openly node-disjoint paths. A corollary is that, for metric costs and for each $k = 1, 2, \ldots, n-1$, there exists a k-node connected graph whose cost is within a factor of 24 of the cost of any simple k-edge connected graph. This resolves an open question in the area. Based on our $O(1)$-approximation algorithm, we present an $O(\log r_{\max})$-approximation algorithm for the metric-cost node-connectivity survivable network design problem where $r_{\max}$ denotes the maximum requirement over all pairs of nodes. Our results contrast with the case of edge costs of zero or one, where Kortsarz et al. (SICOMP **33**, pp.704-720) recently proved, assuming NP$\not\subseteq$ quasi-P, a hardness-of-approximation lower bound of $2^{\log^{1-\epsilon} n}$ for the subset k-node-connectivity problem, where ϵ denotes a small positive number.

1 Introduction

A basic problem in network design is to find a minimum-cost sub-network H of a given network G such that H satisfies some prespecified connectivity requirements. Fundamental examples include the *minimum spanning tree* (MST) problem and the *traveling salesman problem* (TSP). By a *network* we mean an undirected graph together with non-negative costs for the edges, and we use n to denote the number of nodes. Our focus is on networks where the edge costs are metric; that is, the edge costs satisfy the triangle inequalities. This special case is significant from both theoretical and practical viewpoints; metric costs arise in many applications of network design, and perhaps in most of the obvious ones, such as the design of telecommunication networks. Our goal is to design and analyse approximation algorithms for some key problems in network design. Moreover, we resolve a long-standing conjecture on metric graphs, where by a *metric graph* we mean a complete graph K_n together with edge-costs that satisfy the triangle inequalities.

We attack the metric-cost *node-connectivity survivable network design problem* (NC-SNDP). In this problem, we are given a metric graph, as well as a connectivity requirement $r_{i,j}$ between every pair of nodes i and j. Let $r_{\max}$ denote $\max_{i,j \in V} r_{i,j}$. The goal is to find a minimum-cost subgraph H that satisfies these requirements, that is, H should have $r_{i,j}$ openly node-disjoint paths between every pair of nodes i and j. There are two well-known special cases of NC-SNDP. The first is the *subset k-node-connectivity problem*, where we are given a set of terminal nodes $T \subseteq V$ and $r_{i,j} = k$ precisely if both i and j are in T, otherwise $r_{i,j} = 0$. The second is the classical *k-node connected spanning subgraph problem* (k-NCSS) where $r_{i,j} = k$ for every pair of nodes; this is the special case of the subset k-node-connectivity problem with $T = V$. We also study a new special case of NC-SNDP that we call the *subset $[k,\ 1.5k]$-node-connectivity* problem: given a set of terminal nodes $T \subseteq V$ and an (integer) requirement r_i for each node $i \in T$, where $1 \leq k \leq r_i \leq 1.5k$, the goal is to find a minimum-cost subgraph that has $\min(r_i, r_j)$ openly node-disjoint

(28 April 2005)

*Dept. of Comb. & Opt., University of Waterloo, Waterloo, ON, Canada. jcheriyan@uwaterloo.ca

†Department of Mathematics and Statistics, and School of Computer Science, McGill University. vetta@math.mcgill.ca

i,j-paths for every pair of nodes $i,j \in T$. (Thus the subset k-node-connectivity problem is the special case where $r_i = k,\ \forall i \in T$.) See Section 4 for more discussion.

Most network design problems stay NP-hard and APX-hard even assuming metric costs. This remains true even for small connectivity requirements; for example, Bern & Plassmann [3] showed that the Steiner tree problem (the classical special case of the subset k-node-connectivity problem with $k = 1$) is APX-hard even with edge costs of 1 and 2. Over the past decade, there has been significant research on approximation algorithms for network design, and there have been some notable successes in the design of networks that satisfy various types of "edge connectivity" requirements, e.g., Goemans & Williamson [17], and Jain [18], but from the perspective of approximation algorithms, the design of networks subject to "node connectivity" requirements is a murky area. For example, Kortsarz, Krauthgamer & Lee [21] recently proved a hardness-of-approximation lower bound of $2^{\log^{1-\epsilon} n}$ for the subset k-node connectivity problem in graphs with zero-one edge costs, provided that NP$\not\subseteq$ DTIME($n^{polylog(n)}$), where, ϵ denotes a small positive real number. (We give a detailed discussion on previous work in the area after stating our results.)

We present a 24-approximation algorithm for the metric-cost subset k-node-connectivity problem, and then we generalize this to get an $O(1)$-approximation algorithm for the metric-cost subset $[k,\ 1.5k]$-node-connectivity problem. Modulo P$\neq$NP and up to constant factors, these are the best possible results. These algorithms are deterministic and combinatorial; they do not use linear programming relaxations. Based on this, we present an $O(\log r_{\max})$-approximation algorithm for the metric-cost NC-SNDP. The algorithm for NC-SNDP is based on a linear programming relaxation. Also, it uses a 2-approximation algorithm of Goemans & Williamson [17] (see also Agrawal et al. [1]) for the generalized Steiner tree problem. Moreover, we resolve the following long-standing conjecture: In a metric graph and for each $k = 1, 2, \ldots, n-1$, the minimum cost of a k-node connected spanning subgraph is within a constant factor of the minimum cost of a simple k-edge connected spanning subgraph. Thus, for metric graphs, the requirements of k-node-connectivity and simple k-edge-connectivity are equivalent for the objective function, up to constant factors. A similar result holds for requirements of subset $[k,\ 1.5k]$-node-connectivity versus subset simple $[k,\ 1.5k]$-edge-connectivity.

We apply two lower bounds on the optimal value of the subset $[k,\ 1.5k]$-connectivity problem. We may assume (without loss of generality) that there exist at least two terminals with the maximum requirement. Hence, every solution subgraph has at least r_i edges incident to each terminal i, because there is another terminal j with $r_j \geq r_i$, so the solution subgraph must have r_i openly node-disjoint i,j-paths. Our first lower bound comes from the the minimum cost of a subgraph that has degree $\geq r_i$ for every terminal i. Our second lower bound comes from the cost of a minimum spanning tree of the subgraph induced by the terminals. For any node i, we use σ_i or $\sigma(i)$ to denote the sum of the costs of the r_i cheapest edges incident to i in the complete graph, and for any set of nodes S, we use $\sigma(S)$ to denote $\sum_{i \in S} \sigma_i$. We use the abbreviations MST for minimum-cost spanning tree, and TSP for the traveling salesman problem. Let $mst(T)$ denote the cost of an MST of the subgraph induced by T. Our lower bounds are:

(i) $\frac{1}{2}\,\sigma(T)$, and

(ii) $\frac{k}{2}\,mst(T)$.

Note that these lower bounds apply also to the simple edge-connectivity version of the subset $[k,\ 1.5k]$-connectivity problem, where the requirements are for $\min(r_i, r_j)$ edge-disjoint paths between every pair of nodes $i,j \in T$; note that multi-edges are not allowed in the solution subgraph. See Section 2 for more details. Throughout, we use OPT to denote the cost of an optimal solution. Next, we state our main results formally.

Theorem 1 *There is a polynomial-time algorithm for computing a solution to the metric-cost subset k-node connectivity problem of cost $\leq 10\sigma(T) + 4(\frac{k}{2})\ mst(T) \leq 24\text{OPT}$.*

Consider k-NCSS, the special case of the subset k-node connectivity problem in which the terminal set T is V. Let k-ECSS be the problem of finding a minimum-cost *simple* k-edge connected spanning subgraph. Then our two lower bounds apply for both k-NCSS and k-ECSS. This gives the next result.

Corollary 2 *In a network with metric costs, there is a k-node connected spanning subgraph whose cost is at most 24 times the minimum cost of a simple k-edge connected spanning subgraph.*

Remarks: For metric graphs, it is well known that there exists a 2-node connected graph of cost $\leq$ the cost of any 2-edge connected graph (see Appendix 1), but this does not hold for $k \geq 3$ (see [4, Fig.1] and Appendix 1 for examples). Also, note that the $\frac{1}{2}\sigma(V)$ lower bound for k-ECSS does not apply for the version where multi-edges are allowed. In more detail, if multi-edges are allowed, then there exist k-edge connected graphs H such that any k-node connected graph has cost $\geq \Theta(k)\, c(H)$. See Appendix 1 for more details.

Theorem 3 *There is a polynomial-time algorithm for computing a solution to the metric-cost subset $[k,\ 1.5k]$-node-connectivity problem of cost $\leq O(1) \cdot (\sigma(T) + \frac{k}{2}\ mst(T)) \leq O(1) \cdot \text{OPT}$.*

Remark: A loose analysis gives a constant factor between 800 and 1000 in the above theorem. Possibly, an approximation guarantee of ≤ 100 can be obtained by some changes to the algorithm. We have not attempted to optimise the constants in the approximation guarantees.

Theorem 4 *There is a polynomial-time algorithm for computing a solution to the metric-cost NC-SNDP of cost $\leq O(\ln r_{max}) \cdot \text{OPT}$.*

Previous work

Over the past few decades, there has been significant research on approximation algorithms for network design. For early work in network design, see for example Dantzig, Ford & Fulkerson [12]. A celebrated and still unsurpassed result was Christofides' $\frac{3}{2}$-approximation algorithm for the metric-cost TSP [8]. Partly motivated by Christofides' result, there followed a stream of research on related problems in the design of metric-cost networks. Most of this research focused on small connectivity requirements, such as 2-edge connectivity and 2-node connectivity; see Frederickson & Ja'Ja' [14], Monma & Shallcross [26], Monma, Munson & Pulleyblank [25], and Bienstock, Brickell & Monma [4]. For constant k, this last paper gives a constant-factor approximation algorithm for k-NCSS. Moreover, the proof also shows that for metric graphs and any constant k, there exists a k-node connected spanning subgraph of K_n whose cost is within a constant factor of the cost of any k-edge connected spanning subgraph, see [4, Sec.4]. They left open the question of extending these results to all k. This was followed by another burst of research, partly initiated by the work of Goemans & Bertsimas [15] who presented a logarithmic approximation algorithm for a general model called the edge-connectivity survivable network design problem (EC-SNDP) assuming metric costs. Soon after this, the research focus changed from metric costs to the more general setting of non-negative costs. Agrawal, Klein & Ravi [1], and Goemans & Williamson [17] built on the primal-dual method to obtain $O(1)$-approximation algorithms for some special cases of EC-SNDP with small (i.e., zero and one) connectivity requirements. Later, these methods were generalized to EC-SNDP, albeit with a logarithmic approximation guarantee, by Goemans et al. [16] based on work by Williamson et al. [31]. This line of research culminated with a 2-approximation algorithm for EC-SNDP by Jain [18].

Although there was considerable interest in extending these methods to the setting of node connectivity, there was limited success even for rather special cases of NC-SNDP. We mention a few results and refer the interested reader to [6] for more references. For the case of non-negative edge costs, Kortsarz & Nutov [22] and [7] have logarithmic (or worse) approximation guarantees for the k-NCSS problem. For metric costs, there is an $O(1)$-approximation algorithm due to Khuller & Raghavachari [20], and there are other related results in [5, 23]. Some explanation for this lack of good approximation algorithms for NC-SNDP comes from the recent hardness-of-approximation results of Kortsarz, Krauthgamer & Lee [21]. Also, see the surveys by Frank [13], Khuller [19], and Stoer [28], and the book by Vazirani [30].

We briefly mention the relationship between our work and the stream of exciting recent results on PTAS's (polynomial-time approximation schemes) for related problems. Beginning with the results of Arora [2] on the Euclidean TSP, many PTAS's have been obtained for problems in "geometric network design" where the edge costs come from special metrics such as the Euclidean metric, see [9, 10, 11, 27] and the references in those papers. But, modulo P$\neq$NP, such PTAS's do not exist in the setting of interest to us, namely, (general) metric costs; this follows from APX-hardness results in [3, 21, 29].

The rest of the paper is structured as follows. In Section 2, we discuss some preliminaries, and give an overview of our method for the metric-cost subset k-node connectivity problem. We present a constant-factor approximation algorithm for the problem in Section 3. Section 4 gives a constant-factor approximation algorithm for a generalisation. This leads to an $O(\log r_{\max})$-approximation algorithm for the metric-cost NC-SNDP in Section 5.

2 Preliminaries and an overview of the algorithm for subset k-connectivity

Apart from Section 1, we omit the word 'node' from terms such as 'node-connectivity' when there is no danger of ambiguity.

Let the input graph be $G = (V, E)$. We denote the nodes by numbers $i = 1, 2, \ldots, n$, and for nodes i, j the edge between them is denoted ij. The cost of an edge $ij \in E$ is denoted c_{ij} or $c(i, j)$. The costs are said to be *metric* if the triangle inequality holds: $c(v, w) \leq c(v, u) + c(u, w)$, $\forall u, v, w \in V$. Whenever we assume metric costs, we also assume that G is the complete graph. Let k be an integer such that $n > k \geq 1$ (k may be a function of n). For a pair of nodes i, j, let $\kappa(i, j)$ denote the maximum number of openly node-disjoint i, j-paths. Recall that T denotes the set of terminal nodes. We use n' to denote $|T|$, and we assume $T = \{1, \ldots, n'\}$.

Let us formalize the lower bounds (i) and (ii) for the subset $[k,\ 1.5k]$-connectivity problem stated in Section 1. For each terminal node i, let Γ_i denote the set of r_i nearest neighbours of i; by convention, $i \notin \Gamma_i$. (Thus $|\Gamma_i| = r_i$ and $\forall x \in \Gamma_i, y \notin \Gamma_i \cup \{i\}$, $c_{iy} \geq c_{ix}$.) Then note that σ_i denotes $\sum_{x \in \Gamma_i} c_{ix}$. Also, for each terminal node i, let μ_i denote σ_i / r_i, namely, the average cost of an edge from i to one of its r_i nearest neighbours. Note that each terminal node i has at least r_i neighbours in an optimal subgraph, thus $\textsc{opt} \geq \frac{1}{2}\sigma(T)$. This gives the first lower bound. Next, we claim that $\textsc{opt} \geq \frac{k}{2} mst(T)$. In more detail, we have $\textsc{opt} \geq \frac{1}{2}\textsc{ecopt}(T, 2k) \geq \frac{k}{2} mst(T)$, where $\textsc{ecopt}(T, \lambda)$ denotes the minimum cost of a λ-edge connected subgraph of $G[T]$ (allowing multi-edges). To see this, start with a graph corresponding to $\textsc{opt}$, and take two copies per edge to get an Eulerian multi-graph H' that is $2k$-edge connected on T, then apply the Lovász-Mader splitting-off theorem [24, Ex.6.51], [13], to eliminate all nodes of $V - T$ from H' to get a $2k$-edge connected multi-graph on the node set T that has cost $\geq \textsc{ecopt}(T, 2k)$; then we apply the well-known fact that $\textsc{ecopt}(T, \lambda) \geq \frac{\lambda}{2} mst(T)$. For metric costs, splitting off edges does not increase the cost. This gives the second lower bound: $\textsc{opt} \geq \frac{k}{2} mst(T)$.

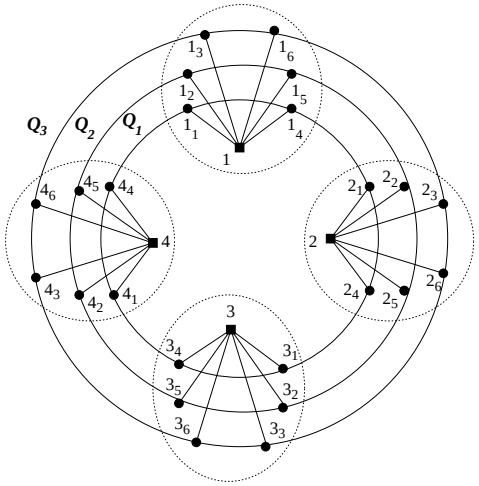

Figure 1: A key special case of the algorithm. Here, $k = 6$, $T = \{1, 2, 3, 4\}$, and the sets $\{i\} \cup \Gamma_i$ (indicated by dotted blobs) for $i \in T$ are pairwise disjoint. The tracks Q_1, Q_2, Q_3 are indicated by circles.

We first give an overview of our method for subset k-connectivity by describing a key special case where k is even, say $k = 2\ell$, and the sets $\{i\} \cup \Gamma_i$ of the terminals i are pairwise disjoint (that is, $(\{i\} \cup \Gamma_i) \cap (\{j\} \cup \Gamma_j) = \emptyset$, $\forall i \neq j \in T$). Arbitrarily name the nodes in Γ_i as $i_1, i_2, \ldots, i_k$, $\forall i \in T$. Construct a cheap cycle Q on the terminals using the well-known MST-doubling heuristic for the TSP. (Start with an MST of the subgraph induced by T, replace each edge by two copies, and shortcut the resulting connected Eulerian graph to get a cycle Q with $V(Q) = T$ and $c(Q) \leq 2mst(T)$.) Let the sequence of terminals on Q be $1, 2, \ldots, n', 1$ (renumber the nodes if needed). For each $\tau = 1, \ldots, \ell$, construct a cycle Q_τ "parallel" to Q where $Q_\tau = 1_\tau, 1_{\ell+\tau}, 2_\tau, 2_{\ell+\tau}, 3_\tau, \ldots, (n'-1)_{\ell+\tau}, n'_\tau, n'_{\ell+\tau}, 1_\tau$. (See Figure 1; informally, start with the cycle $1_\tau, 2_\tau, \ldots, n'_\tau, 1_\tau$, then for each $i = 1, \ldots, n'$ insert the node $i_{\ell+\tau}$ between nodes i_τ and $(i+1)_\tau$.) Let us refer to these cycles as *tracks*. It can be seen that a track Q_τ has cost $c(Q_\tau) \leq c(Q) + \sum_{i=1}^{t} 2(c(i, i_\tau) + c(i, i_{\ell+\tau}))$ (see the second subroutine below), and the total cost of the tracks is $\sum_{\tau=1}^{\ell} c(Q_\tau) \leq \ell \cdot c(Q) + 2\sigma(T)$. Finally, for each terminal $i \in T$, we add the k edges $ii_1, ii_2, \ldots, ii_k$. The resulting subgraph is our solution graph H; it has cost $c(H) \leq 2\ell \cdot mst(T) + 3\sigma(T) \leq 2\text{OPT} + 6\text{OPT} = 8\text{OPT}$. Note that each terminal has precisely two neighbours in each track. Thus H satisfies the connectivity requirements, because for every pair of terminals $i, j (i \neq j)$, each of the $k/2$ tracks contributes 2 openly disjoint i, j paths.

The algorithm uses the following two subroutines. Note that the solution graph H is simple, so when we add edges to H we do so without creating multi-edges.

- The first subroutine *copies a specified set of neighbours* of a terminal i to another terminal v (possibly, v is adjacent to i). More precisely, given a terminal i and a specified set of neighbours of i, call it N_i, and another terminal v, the subroutine adds an edge vx to H for each node $x \in N_i$ (without creating multi-edges or loops in H). After this step, $\kappa(i, v) \geq |N_i|$ in H. The cost of the new edges is $\leq |N_i|\ c(i, v) + \sum_{x \in N_i} c(i, x)$; moreover, if there is a positive real number γ such that $\sum_{x \in N_i} c(i, x) \leq \gamma \sigma_i$, then the cost of the new edges is $\leq |N_i|\ c(i, v) + \gamma \sigma_i$.

- The other subroutine starts with a cycle containing a terminal i and *inserts new node(s) into the cycle*. Given a cycle Q', a terminal i in Q', and a node $x \notin V(Q')$, we first add two copies of the edge ix to Q' to get a connected Eulerian graph. Then we shortcut this Eulerian graph (as in the

MST-doubling heuristic for the TSP) to obtain a new cycle Q with node set $V(Q') \cup \{x\}$. The increase in cost is $\leq 2c(i,x)$.

It is important for our analysis to get good upper bounds on the costs of the tracks. Note that the tracks are pairwise node disjoint; thus each terminal is in at most one track. But, for upper-bounding the track costs, we use the following accounting trick: Consider any track Q_τ. We assume that the track initially consists of all the terminals, thus $V(Q_\tau) = T$, and using the MST-doubling heuristic we have $c(Q_\tau) \leq 2mst(T)$. Subsequently, the algorithm may insert new nodes into the track – such insertions occur while we are processing some terminal – thus for inserting node x while processing terminal i the cost $c(Q_\tau)$ increases by $\leq 2c(i,x)$. Possibly, x may be another terminal – in that case, we implicitly remove x from Q_τ and then insert x via the double-edge ix. At the end of the execution, we keep only those terminals that were explicitly inserted into Q_τ and remove all the other terminals from Q_τ; clearly, this does not increase the cost $c(Q_\tau)$. Note that this "historical view" of Q_τ is only needed for upper-bounding the cost. Other than this, it may be easier to view the tracks as being pairwise node disjoint all through the execution, and this is the viewpoint we use in presenting the detailed algorithm.

3 The algorithm for subset k-connectivity

This section is devoted to an algorithm and proof for Theorem 1. The detailed algorithm follows. An analysis of the cost of the edges added to H (the solution graph) is given after the algorithm. A terminal may be in two states *active* or *inactive*. Initially, all the terminals are active. Let ℓ denote $\lceil k/2 \rceil$. Initially, H is the graph consisting of all the terminal nodes and no edges, thus $H = (T, \emptyset)$.

(1) [DE-ACTIVATE TERMINALS & CONSTRUCT DISJOINT BALLS FOR ACTIVE TERMINALS]
Renumber the terminals as $1, 2, \ldots, n'$ by increasing value of μ; thus $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{n'}$.
Note: $\mu_h \leq \mu_j$ iff $\sigma_h \leq \sigma_j$.
Scan the terminals in the order $1, 2, \ldots, n'$, and skip the current terminal if it is inactive. For an active terminal i, construct the set $B_i = \{j \mid c(i,j) \leq \alpha\mu_i\}$, where we choose $\alpha = 2$. For each active terminal $v > i$, if $c_{iv} \leq (\alpha\mu_i + \beta\alpha\mu_v)$, where we fix $\beta = 2$, then make v inactive, and record i as the parent of v by assigning $p(v) = i$. (The aim is to ensure that the sets B_i of active terminals i are pairwise disjoint.)

Note that $i \in B_i$ and $|B_i| \geq 1 + (1 - \frac{1}{\alpha})k = 1 + \frac{k}{2}$. (Otherwise, we have $\geq k/\alpha = k/2$ nodes x in Γ_i with $c(i,x) > \alpha\mu_i = \alpha\sigma_i/k$, so these nodes contribute $> \sigma_i$ to $\sum_{x \in \Gamma_i} c(i,x)$.) Hence, $|B_i - \{i\}| \geq \ell$. Also note that $\mu_{p(v)} \leq \mu_v$ for each inactive terminal v.

Choose the ℓ nodes in B_i nearest to i and name them as $i_1, i_2, \ldots, i_\ell$ such that $c(i,i_1) \leq c(i,i_2) \leq \cdots \leq c(i,i_\ell)$.

(2) [CONSTRUCT ℓ TRACKS ON THE DISJOINT BALLS]
After step (1), let T^* denote the set of active terminals and let $n^* = |T^*|$. If $n^* < 3$, then apply step (2′) and stop. Otherwise, construct a cheap cycle Q on the active terminals by applying the MST-doubling heuristic for the TSP to the subgraph induced by T^*. Renumber the terminals such that $Q = 1, 2, \ldots, n^*, 1$, that is, the active terminals get the numbers in $\{1, \ldots, n^*\}$ according to their ordering in Q. Construct ℓ tracks $Q_1, Q_2, \ldots, Q_\ell$, where track $Q_\tau = 1_\tau, 2_\tau, \ldots, n^*_\tau, 1_\tau (\tau = 1, \ldots, \ell)$. Add all the tracks (but not the cycle Q) to H. The cost of the tracks constructed in this step is analysed in Proposition 6 below.

(2′) [SPECIAL HANDLING FOR 1 OR 2 ACTIVE TERMINALS]
Skip this step if $n^* \geq 3$. Suppose $n^* = 1$. Let the active terminal be i. Add all the edges $iv, v \in \Gamma_i$, and then for each inactive terminal j, copy the set Γ_i of neighbours of i to j. The resulting graph H satisfies the connectivity requirements.

Suppose $n^* = 2$. Let the active terminals be h, i, with $\sigma_h \leq \sigma_i$. Add all the edges $hq, q \in \Gamma_h$, and $iv, v \in \Gamma_i$. Then add a matching M of maximum size between the nodes in $\Gamma_i - (\Gamma_h \cup \{h\})$ and in $\Gamma_h - (\Gamma_i \cup \{i\})$; now, each matching edge qv (say $q \in \Gamma_h - \{i\}$ and $v \in \Gamma_i - \{h\}$) gives an h, i path, namely, h, q, v, i. Finally, for each inactive terminal j, copy the set $\Gamma_{p(j)}$ of neighbours of $p(j)$ to j. The resulting graph H satisfies the connectivity requirements.

(3) [AUGMENT DISJOINT BALLS AND ASSIGN TOKEN ARCS]
In summary, this step scans the active terminals i and augments each "ball" B_i to get an "augmented ball" B'_i (that ideally has $|B'_i| \geq r_i + 1 = k + 1$) such that these augmented balls are pairwise disjoint. The obvious construction for B'_i is to start with B_i and then add some nodes from $\Gamma_i - B_i$, but then the augmented balls may intersect. We "de-intersect" two intersecting sets B'_h and B'_i, while preserving the balls B_h and B_i, by assigning so-called *token arcs* to the active terminals such that for each active terminal i, $|B'_i|$ plus the number of token arcs assigned to i is $\geq r_i + 1 = k + 1$. Consider one special case: suppose that h and i are active terminals and node q is in $B'_h \cap B'_i$ but $q \notin B_h \cup B_i$. Then we compare the costs of the edges hq and iq and "replace" the costlier edge, say iq, by a token arc whose cost we fix to be $3c_{iq}$; that is, we remove q from B'_i and instead assign to i a token arc with cost $3c_{iq}$. The details follow.

Renumber the terminals so that the active terminals i in order of increasing μ_i values are $1, 2, \ldots, n^*$, and scan the active terminals in this order. Start the scan of $i \in T^*$ by defining $B'_i := B_i$ if $\Gamma_i \subseteq B_i$, and $B'_i := \Gamma_i$ otherwise. If B'_i is disjoint from B'_h for all active terminals $h < i$, then continue with the next active terminal, otherwise, for each active terminal $h < i$ with $B'_h \cap B'_i \neq \emptyset$, examine the nodes q in $B'_h \cap B'_i$ in any order. Note that $\mu_h \leq \mu_i$.

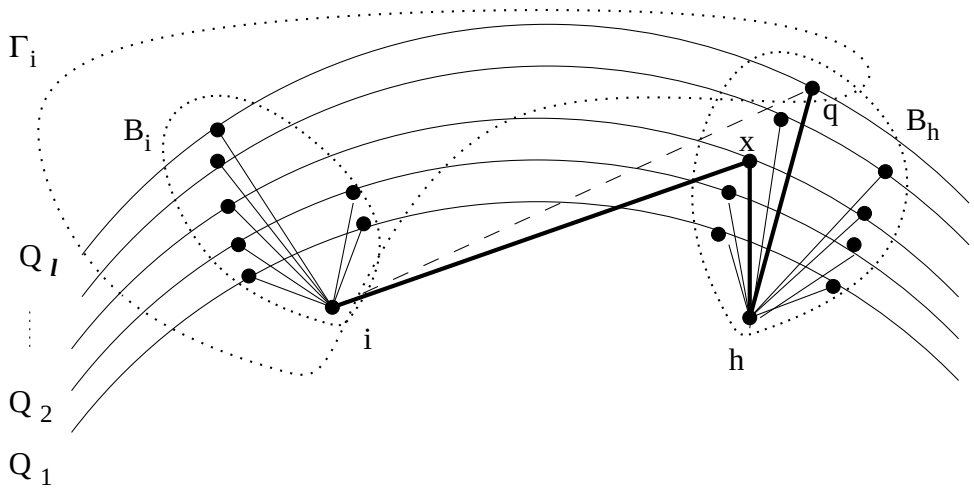

Figure 2: An illustration of step (3)(a) in Section 3: the "dashed edge" iq is replaced by a token arc ih that is later (in step (4)) replaced by an edge ix, $x \in B_h$.

(a) Suppose $q \in B_h$. Then note that $q \notin B_i$ and $c_{iq} > \alpha\mu_i \geq \alpha\mu_h$. Remove q from B'_i and give to i a token arc (i, h) with cost $3c_{iq}$. (Later, this token arc will be replaced by an edge ix where

$x \in B_h$; note that the cost of ix is $\leq c_{iq} + c_{hq} + c_{hx} \leq c_{iq} + 2\alpha\mu_h \leq 3c_{iq}$.) See Figure 2 for an illustration.

(b) Otherwise, $q \in B'_h - B_h$. Suppose $q \in B_i$. Then note that $c_{hq}+c_{iq} \geq c_{hi} \geq (\alpha\mu_h+\beta\alpha\mu_i)$ (the last inequality holds because both h, i are active), and $c_{iq} \leq \alpha\mu_i$, hence, $c_{hq} \geq \alpha(\mu_h+\mu_i)$ (recall that $\beta = 2$). Remove q from B'_h and give to h a token arc (h, i) with cost $3c_{hq}$. (Later, this token arc will be replaced by an edge hx, $x \in B_i$, of cost $\leq c_{hq}+c_{iq}+c_{ix} \leq c_{hq}+2\alpha\mu_i \leq c_{hq}+2c_{hq} \leq 3c_{hq}$.)

(c) Suppose $q \in B'_h - B_h$ and $q \in B'_i - B_i$. Then we compare c_{iq} and c_{hq}.

If $c_{iq} \geq c_{hq}$, then remove q from B'_i and give to i a token arc (i, h) with cost $3c_{iq}$. (Later, this token arc will be replaced by an edge ix, $x \in B_h$, of cost $\leq c_{iq} + c_{hq} + c_{hx} \leq 2c_{iq} + \alpha\mu_h \leq 3c_{iq}$, where the last inequality holds because $c_{iq} > \alpha\mu_i \geq \alpha\mu_h$.)

Otherwise, we have $c_{iq} < c_{hq}$. Then we remove q from B'_h and give to h a token arc (h, i) with cost $3c_{hq}$. (Later, this token arc will be replaced by an edge hx, $x \in B_i$, of cost $\leq c_{hq}+c_{iq}+c_{ix} \leq 2c_{hq} + \alpha\mu_i \leq 3c_{hq}$, where the last inequality holds because $c_{hq} + c_{iq} \geq c_{hi} \geq (\alpha\mu_h + \beta\alpha\mu_i)$ (as in (b) above), hence, $c_{hq} \geq \frac{\alpha}{2}(\mu_h + \beta\mu_i) \geq \alpha\mu_i$ (recall that $\beta = 2$).)

After step (3), note that the cost of a token arc (i, j) depends on the cost of the associated edge iq and is $3c_{iq}$.

(4) [ATTACH ACTIVE TERMINALS TO TRACKS]
In summary, this step scans each active terminal i and adds edges from i to the tracks such that each track $Q_\tau, \tau = 1, \ldots, \lfloor k/2 \rfloor$, gets two neighbours of i, and the last track Q_ℓ gets ≥ 1 neighbour of i.

First add edges from i to each of $i_1, i_2, \ldots, i_\ell$; also, mark the nodes $i_1, i_2, \ldots, i_\ell$ as used.

Then for each $\tau = 1, 2, \ldots, \lfloor k/2 \rfloor$, do the following. If an unused token arc (i, h) is available, then choose it, mark it as used, and add the edge ih_τ; note that h_τ is in B_h and is the "first neighbour" of h in track Q_τ; also, note that $c(i, h_\tau)$ is $\leq$ the cost of the token arc (i, h). If no unused token arcs are available, then choose an unused node $q \in B'_i$, mark it as used, insert q into track Q_τ, and add the edge iq. (Note that the number of token arcs given to i plus $|B'_i|$ is $\geq k + 1$, hence, this step will find $\lfloor k/2 \rfloor$ token arcs or unused nodes, excluding the nodes $i_1, i_2, \ldots, i_\ell$.)

For each active terminal i, let N_i denote the set of neighbours of i in the tracks, just after step (4) is applied to i.

(5) [ATTACH INACTIVE TERMINALS TO TRACKS]
Finally, "attach" the inactive terminals to the tracks. Note that an inactive terminal may be already in one of the tracks. For each inactive terminal j, copy the set of neighbours $N_{p(j)}$ of the parent $p(j)$ to j.

Proposition 5 *The cost of the graph constructed in step (2′) is* ≤ 16OPT.

Proof: Suppose $n^* = 1$, and let i be the (unique) active terminal. Then $c(H) \leq \sigma_i + \sum_{j \in T-T^*}(kc_{ij} + \sigma_i) \leq \sigma_i + \sum_{j \in T-T^*}(k(\alpha\mu_i + \beta\alpha\mu_j) + \sigma_i) \leq \sigma_i + \sum_{j \in T-T^*}(\alpha(1+\beta)\sigma_j + \sigma_j) \leq 7\sigma(T) \leq 14$OPT (we have $\alpha = 2, \beta = 2$, and we used $\sigma_i \leq \sigma_j$ for an inactive terminal j).

Suppose $n^* = 2$, and let i, h be the two active terminals. Then recall that M denotes a matching of maximum size between the nodes in $\Gamma_i - (\Gamma_h \cup \{h\})$ and in $\Gamma_h - (\Gamma_i \cup \{i\})$; note that an edge $qv \in M$ (say, $q \in \Gamma_h, v \in \Gamma_i$) has cost $\leq c_{hq} + c_{hi} + c_{iv}$, hence, $c(M) \leq \sigma_h + \sigma_i + k \cdot mst(T)$; the other edges in H contribute a cost of $\leq \sigma_h + \sigma_i + \sum_{j \in T-T^*}(\alpha(1+\beta)\sigma_j + \sigma_j)$ (as in the analysis for $n^* = 1$) hence, $c(H) \leq 7\sigma(T) + k \cdot mst(T) \leq 16$OPT. ■

Proposition 6 *(i) The total cost of the edges added by step (4) and incident to an active terminal i is $\leq 4\sigma_i$. (ii) At the end of step (4), the total cost of the ℓ tracks is $\leq 2\ell \cdot mst(T) + 4\sigma(T^*)$.*

Proof: For an active terminal i, the total cost of the token arcs (i,h) given to i is $\leq 3\sigma_i$. The cost of the edges added that are incident to i, but excluding the cost due to the token arcs, is $\leq \alpha\sigma_i$ if $|B_i| \geq k+1$ (in this case, $B'_i = B_i$ and no token arcs are given to i), and is $\leq \sum_{x\in\Gamma_i} c_{ix} \leq \sigma_i$ otherwise. Thus the total cost of the added edges incident to i is $\leq \max(\alpha\sigma_i,\ \sigma_i + 3\sigma_i) \leq 4\sigma_i$.

The total cost of the ℓ tracks (that were constructed in step (2) and modified in step (4)) is $\leq 2\ell \cdot mst(T) + 4\sigma(T^*)$. To see this, first consider the term $2\ell \cdot mst(T)$. Recall (from Section 2) the accounting trick we use for upper-bounding the cost of a track; due to this, we take the upper bound on the cost of Q (the cheap cycle on T^* in step (2)) to be $2mst(T)$ rather than $2mst(T^*)$. Summed over ℓ tracks, this gives $2\ell \cdot mst(T)$. For the second term, note that $i \in T^*$ contributes $\leq \sum_{q\in B'_i} 2c(i,q)$, and this is $\leq 2k(\alpha\mu_i)$ if $\Gamma_i \subseteq B_i$ (then $B'_i = B_i$), and $\leq 2\sigma_i$ otherwise (then $B'_i = \Gamma_i$). ■

Proposition 7 *The total cost of the edges added by step (5) and incident to the inactive terminals is $\leq 10\sigma(T - T^*)$.*

Proof: Suppose the cost of the added edges incident to an active terminal i is $\leq \gamma\sigma_i$. (From Proposition 6, we have $\gamma = 4$.) Then the cost of the edges added for an inactive terminal j with parent i is $\leq k \cdot c_{ij} + \gamma\sigma_i \leq k(\alpha\mu_i + \beta\alpha\mu_j) + \gamma\sigma_i \leq (\alpha(\beta+1)+\gamma)\sigma_j$, using the fact that $\sigma_{p(j)} \leq \sigma_j$. Thus the total cost of the edges added in this step is $\leq 10\sigma(T - T^*)$, using $\alpha = 2, \beta = 2, \gamma = 4$. ■

Proof of Theorem 1: By the above propositions, the total cost of H is $\leq 2\ell \cdot mst(T) + 4\sigma(T^*) + \gamma\sigma(T^*) + 10\sigma(T - T^*) \leq (k+1)mst(T) + 8\sigma(T^*) + 10\sigma(T - T^*) \leq (k+1)mst(T) + 10\sigma(T) \leq (2 + \frac{2}{k})\text{OPT} + 20\text{OPT} \leq 24\text{OPT}$.

We claim that the graph H has the required connectivity property, namely, $\kappa(i,j) \geq k, \forall i \neq j \in T$. To see this, consider any pair of terminals i, j and consider any one track Q_τ. Suppose that either i is in Q_τ, or i is not in Q_τ but has two neighbours in Q_τ. Suppose the same statement holds for j (that is, j is in Q_τ, or j is not in Q_τ but has two neighbours in Q_τ). Then, Q_τ (together with the edges from i and j to Q_τ) contributes two openly disjoint i,j paths. Similarly, Q_τ contributes one i,j path if both i and j either are in Q_τ or have a neighbour in Q_τ. By construction, each active terminal has two neighbours in each of the tracks Q_τ for $\tau = 1, \ldots, \lfloor k/2 \rfloor$, and has a neighbour in Q_ℓ; similarly, each inactive terminal is either in Q_τ or has two neighbours in Q_τ for $\tau = 1, \ldots, \lfloor k/2 \rfloor$, and is in Q_ℓ or has a neighbour in Q_ℓ. Then, for any two terminals i and j, H has k openly disjoint i,j paths, since each of the tracks Q_τ for $\tau = 1, \ldots, \lfloor k/2 \rfloor$, contributes two openly disjoint i,j paths, Q_ℓ contributes an i,j path, and these k paths together are openly disjoint. ■

4 The algorithm for subset $[k, 1.5k]$-connectivity

In this section, we extend the methods of the previous section to obtain an $O(1)$-approximation algorithm for the the subset $[k, 1.5k]$-connectivity problem. It seems likely that these methods will give similar results for the subset $[k, \rho\, k]$-connectivity problem, for any constant ρ, $1 \leq \rho < 2$, but they do not extend to $\rho = 2$ for the following reason: as in Section 3, we choose some terminals to be active and we construct pairwise-disjoint sets B_i of radius $O(1)\mu_i$ for the active terminals i, where B_i has at least a fraction ϕ of the nodes in Γ_i ($\phi = \frac{1}{2}$ in Section 3); our method assumes $\phi \geq \frac{\rho}{2}$, i.e., the size of every set $B_i - \{i\}$ is at least half the maximum requirement; then, for $\rho = 2$ and an active terminal i with $r_i = k$ we need $|B_i - \{i\}| \geq k = |\Gamma_i|$

and this is not possible for sets of radius $O(1)\mu_i$. Our main application is to the NC-SNDP, and for this, any constant $\rho > 1$ suffices; we chose $\rho = 1.5$ for convenience.

The main difficulty in extending the methods of Section 3 comes from the fact that an active terminal i may have an inactive terminal v with $r_v > r_i$ as a child. Then we cannot satisfy the connectivity requirement of v by copying the neighbours of i to v. Roughly speaking, we handle this as follows: we pick a child v^* of i with the maximum requirement, and copy all the neighbours of i to v^*; then, if needed, we add new neighbours for v^* in the tracks by examining the nodes $x \in \Gamma_{v^*}$; if $x \in B'_h$ for some active terminal h then we proceed similarly to step (3) of Section 3 (though there are new complications), otherwise, we either insert x as a new node into a track or we "transform" to the case of $x \in B'_h$. For any other inactive child v of i, we attempt to copy the "first" r_v neighbours of v^* to v. This is an informal (and inaccurate) overview; the details are given below. The main contribution of this section is an algorithm and proof for the following restricted case of Theorem 3.

Theorem 8 *Let k be an integer multiple of 4, thus $k = 0 \pmod 4$. There is polynomial-time algorithm for computing a solution to the metric-cost subset $[k,\ 1.5k]$-connectivity problem of cost $\leq O(1) \cdot \text{OPT}$.*

Remark: A loose analysis gives a constant factor between 800 and 900 in the above theorem.

Theorem 3 follows by combining this theorem with Theorem 1. To see this, suppose that $k \neq 0 \pmod 4$ (otherwise, we are done). Let $\hat{k} \geq k$ denote the next integer multiple of 4; clearly, $\hat{k} - k \leq 3$. Then for each $\rho = k, k+1, \ldots, \hat{k}-1$, we apply the algorithm in Theorem 1 to the following instance $\Pi(\rho)$ of the subset ρ-connectivity problem to obtain a solution subgraph $H(\rho)$: we take the requirement of a terminal i in $\Pi(\rho)$ to be $r'_i = 0$ if $r_i < \rho$, and we take $r'_i = \rho$ if $r_i \geq \rho$; the rest of the instance stays the same. Finally, we apply the algorithm of this section to the instance of subset $[\hat{k},\ 1.5\hat{k}]$-connectivity where we take the requirement of a terminal i to be $r'_i = 0$ if $r_i < \hat{k}$, and we take $r'_i = r_i$ if $r_i \geq \hat{k}$; the rest of the instance stays the same. Let H' be the solution subgraph. Then, for the original instance (of subset $[k,\ 1.5k]$-connectivity), we output the solution subgraph $H^* = H(k) \cup H(k+1) \cup \cdots \cup H(\hat{k}-1) \cup H'$ whose cost is at most $O(1)\text{OPT}$. To see that H^* satisfies the connectivity requirements, note that for every pair of terminals i, j, one of the subgraphs forming H^* (namely, one of $H(k), H(k+1), \ldots, H(\hat{k}-1), H'$) has $\min(r_i, r_j)$ openly disjoint i,j-paths.

Assume that k is an integer multiple of 4. Let ℓ denote $3k/4$. For any terminal i and any edge ix of the complete graph, let $\widetilde{c}_{ix} = \widetilde{c}(i,x)$ denote the normalized edge cost $\max(c_{ix}, \mu_i)$.

The detailed algorithm follows. A terminal may be in two states *active* or *inactive*. Initially, all the terminals are active, and H is the graph consisting of all the terminal nodes and no edges, thus $H = (T, \emptyset)$. See Appendix 2 for a summary of the notation.

(1) [DE-ACTIVATE TERMINALS & CONSTRUCT DISJOINT BALLS FOR ACTIVE TERMINALS]
Renumber the terminals as $1, 2, \ldots, n'$ by increasing value of μ; thus $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{n'}$.
Note: $\mu_h \leq \mu_j$ does not imply $\sigma_h \leq \sigma_j$ since the requirements may differ, but we do have $\sigma_h \leq 1.5\sigma_j$.
Scan the terminals in the order $1, 2, \ldots, n'$, and skip the current terminal if it is inactive. For an active terminal i, construct the set $B_i = \{j \mid c(i,j) \leq \alpha\mu_i\}$, where we choose $\alpha = 4$. For each active terminal $v > i$, if $c_{iv} \leq (\alpha\mu_i + \beta\alpha\mu_v)$, where we fix $\beta = 2$, then make v inactive, and record i as the parent of v by assigning $p(v) = i$. (The aim is to ensure that the sets B_i of active terminals i are pairwise disjoint.)

Note that $i \in B_i$ and $|B_i| \geq 1 + (1 - \frac{1}{\alpha})k = 1 + \frac{3k}{4}$. (Otherwise, we have $\geq k/\alpha = k/4$ nodes x in Γ_i with $c(i,x) > \alpha\mu_i = \alpha\sigma_i/k$, so these nodes contribute $> \sigma_i$ to $\sum_{x \in \Gamma_i} c(i,x)$.) Also note that $\mu_{p(v)} \leq \mu_v$ for each inactive terminal v.

Choose the ℓ nodes in B_i nearest to i and name them as $i_1, i_2, \ldots, i_\ell$ such that $c(i,i_1) \leq c(i,i_2) \leq \cdots \leq c(i,i_\ell)$.

(2) [CONSTRUCT ℓ TRACKS ON THE DISJOINT BALLS]
After step (1), let T^* denote the set of active terminals and let $n^* = |T^*|$. If $n^* < 3$, then apply step (5′) and stop. Otherwise, construct a cheap cycle Q on the active terminals by applying the MST-doubling heuristic for the TSP to the subgraph induced by T^*. Renumber the terminals such that $Q = 1, 2, \ldots, n^*, 1$, that is, the active terminals get the numbers in $\{1, \ldots, n^*\}$ according to their ordering in Q. Construct ℓ tracks $Q_1, Q_2, \ldots, Q_\ell$, where track $Q_\tau = 1_\tau, 2_\tau, \ldots, n^*_\tau, 1_\tau (\tau = 1, \ldots, \ell)$. Moreover, we have a special track $Q_0 = Q$; this track is used to satisfy the requirements of inactive terminals, but not the requirements between active terminals. Add all the tracks to H. (Although the tracks are similar to each other, our construction distinguishes between the tracks and relies on the ordering of the tracks given by the track indices $0, 1, 2, 3, \ldots$.)

(3) [AUGMENT DISJOINT BALLS AND ASSIGN TOKEN ARCS]
This step is the same as step (3) in the algorithm for subset k-connectivity in Section 3, except that some parameters are different: here, we have $\alpha = 4$, $\beta = 2$, $\ell = \frac{3k}{4}$.

After step (3), note that the sets B'_i of the active terminals i are pairwise disjoint, each such set has size $\geq 1 + (3k/\alpha) = 1 + (3k/4)$ (since $B'_i \supseteq B_i$ and $|B_i| \geq 1 + (3k/\alpha)$), and for each active terminal i the number of token arcs given to i plus $|B'_i|$ is $\geq r_i + 1$. Also, note that the cost of a token arc (i,j) depends on the cost of the associated edge iq and is $3c_{iq}$.

(4) [ATTACH ACTIVE TERMINALS TO TRACKS]
In summary, this step scans each active terminal i and adds edges from i to the tracks such that i has a neighbour i_τ in each of the tracks $Q_\tau, \tau = 1, \ldots, \ell$, and moreover, i has a second neighbour in each of the $r_i - \ell$ tracks $Q_\tau, \tau = 1, \ldots, r_i - \ell$. (Possibly, i may have more than $r_i - \ell$ tracks that each have two neighbours of i.) We call i_τ the *inner neighbour* of i in Q_τ, and if i has another neighbour x in Q_τ then we call x the *outer neighbour* of i in Q_τ.

Note that the sets B'_j of the active terminals j are pairwise disjoint. In step (4), every node added to a track is in B'_j for some active terminal j (this can be seen from the description below). We call a node x *free* if $x \notin \bigcup\{B_j \mid j \in T^*\}$ and x is in none of the tracks of the current graph H. While processing a terminal v we may find a free node $x \in \Gamma_v$ and we may insert x as the outer neighbour of v in a track. Throughout the execution, x stays in the same track, and stays as the outer neighbour of v, but other terminals too may add x as their outer neighbour on that track.

We examine the active terminals in any order. Let i be the current active terminal. First, we add edges from i to each of $i_1, i_2, \ldots, i_\ell$; also, we mark the nodes $i_1, i_2, \ldots, i_\ell$ as used (with respect to i). We start with the variable $\tau = 1$; this variable denotes the number of the track where the next outer neighbour of i is placed.

If any unused nodes remain in B_i, then choose an unused node $x \in B_i$ with minimum $c(i,x)$, mark x as used w.r.t. i, insert x into track Q_τ, increase τ by one, and add the edge ix. We repeat this step until either B_i has no unused nodes or $\tau = \ell + 1$ (meaning that i has an outer neighbour in each of the ℓ tracks). If $\tau \geq (r_i - \ell) + 1$, then we are finished with step (4) for i, otherwise, we continue.

If an unused token arc (i,h) is available, then we choose it, mark it as used, add the edge ih_τ, and increase τ by one; note that h_τ is in B_h and is the inner neighbour of h in track Q_τ; also, note that $c(i,h_\tau)$ is $\leq$ the cost of the token arc (i,h). We repeat this step until there are no unused token arcs or $\tau = (r_i - \ell) + 1$. If $\tau < (r_i - \ell) + 1$, then we continue, otherwise, we are finished with step (4) for i.

We choose an unused node $x \in B'_i - B_i$ with minimum $c(i,x)$, and mark it as used w.r.t. i (note that x is a free node). Then we insert x into track Q_τ and add the edge ix, *provided* there exists *no* suitable "target terminal" $h \neq i$ (the details are given below; note that the target terminal is defined with respect to the edge ix). If a suitable h exists, then we discard x and add the edge ih_τ, that is, we take the inner neighbour of h in Q_τ to be the outer neighbour of i in Q_τ. (The reason for using an edge ih_τ rather than ix is that x is a free node now, but later we may find that x is essential for attaching some inactive terminal v to the tracks, and at that step, we will be forced to "replace" the edge ix by some other edge iy; to avoid such "replacements" we look ahead, and we use the edge ix only if there are no "future conflicts" for x.)

The details are as follows. We check whether there exists an active terminal $h \neq i$ such that

$$\text{hop-rule} \qquad c(i,h) \leq (2+(\beta+1)\alpha)c(i,x) \leq 14c(i,x), \qquad \text{and} \qquad \mu_h \leq c(i,x).$$

If such an h exists, then we add the edge ih_τ. Note that $c(i,h_\tau) \leq c(i,h)+\alpha\mu_h \leq (2+(\beta+2)\alpha)c(i,x) \leq 18c(i,x)$. If no such h exists, then (as mentioned before) we insert x into track Q_τ and add the edge ix. In either case, we increase τ by one.

Note that the number of token arcs given to i plus $|B'_i|$ is $\geq r_i+1$, hence, this step will find r_i token arcs or unused nodes (including the nodes $i_1,\ldots,i_\ell$).

After all active terminals have been examined by step (4), it can be seen that H satisfies the connectivity requirements of all active terminals.

Note that an inactive terminal may be in one of the tracks $Q_1, Q_2, \ldots, Q_\ell$, although none of the active terminals is in those tracks.

For each active terminal i, let N_i denote the (ordered) set of neighbours of i in the graph $H - V(Q_0)$ at the end of step (4). (Thus, N_i is the set of neighbours attaching i to the other tracks — excluding the track Q_0 containing i.) Note that $|N_i| \geq r_i$, $\forall i \in T^*$ (if $|B_i| > r_i+1$ then N_i may have $> r_i$ nodes). Moreover, we order the nodes in each set N_i such that the inner neighbours of i come first in the order $i_1, i_2, \ldots, i_\ell$, followed by the outer neighbours in the order of their track numbers (the outer neighbour in Q_1, followed by the outer neighbour in Q_2, ...).

Remark: The ordered sets N_i for $i \in T^*$ are used in step (5), and there it is critical that the total cost of the edges from i to the nodes in N_i is $\leq \gamma\sigma_i$ for a constant γ; in particular, none of these edge costs should be "charged" to the *mst* lower bound.

(5) [ATTACH INACTIVE TERMINALS TO TRACKS]
Finally, "attach" each inactive terminal to the tracks.

By a *sibling* of an inactive terminal v we mean either the parent $p(v)$ or another child of $p(v)$. In summary, we first copy to v the neighbours of a sibling, and then, if needed, we add additional neighbours via Γ_v – note that v's requirement r_v may be much greater than that of any of its siblings, hence, copying the neighbours of a sibling may not suffice. We also use the special track Q_0 to satisfy the requirements of inactive terminals; to see the need for Q_0 consider a child v of an active terminal i with $r_v = r_i+1$ and $\Gamma_v = \{i\} \cup \Gamma_i$; we handle the requirement of v by adding the edge vi – thus attaching v to Q_0 – and then copying the neighbours of i to v.

Focus on an active terminal i and its children, and let v^* have the maximum requirement among these terminals; assume that $v^* \neq i$ (the other case is easy). Step (5) attaches v^* to the tracks via a neighbour in each of the $\ell+1$ tracks $Q_0, Q_1, \ldots, Q_\ell$ and two neighbours in each of the $r_{v^*} - (\ell+1)$

tracks $Q_1, Q_2, \ldots, Q_{r_{v^*}-(\ell+1)}$. These neighbours of v^* constitute the ordered set N_{v^*}; we use our standard ordering, i.e., the neighbour $i = p(v^*)$ in Q_0 comes first, followed by the inner neighbours in the tracks $Q_1, \ldots, Q_\ell$, followed by the outer neighbours, and further, the neighbours are ordered by their track number. Similarly, we have an ordered set of neighbours N_v for each inactive terminal v, where N_v is the (ordered) set of nodes x such that step (5) – while processing v – adds an edge from v to x. (Possibly, v occurs in a track, but then neither of the two neighbours of v in the track occurs in N_v unless step (5) – while processing v – added the edge from v to that node.) A key property of our construction is that for each sibling v of v^*, N_v is a prefix of N_{v^*}; in particular, for each $\tau \in \{1, 2, \ldots, r_v - (\ell+1)\}$, the outer neighbour of v in track Q_τ is the same as the outer neighbour of v^* in that track. Therefore, if Γ_v contains two or more outer neighbours of siblings of v, then all of these outer neighbours are in distinct tracks. (Thus, for siblings $v_1, v_2, \ldots$, our construction makes the sets $N_{v_1}, N_{v_2}, \ldots$ "consistent" even though the sets $\Gamma_{v_1}, \Gamma_{v_2}, \ldots$ may have arbitrary intersections.)

We examine the active terminals i in order of increasing μ_i values, and we examine the children v (of i) in order of increasing μ_v values. By a *prior sibling* of v we mean either the parent $p(v)$ or another child of $p(v)$ that precedes v in this ordering. For each child v of i, define the source terminal of v, denoted $\hat{p}(v)$, to be a prior sibling with the maximum requirement; furthermore, define the ordered set N_v^0 to be $\{p(v)\} \cup N_{p(v)}$ if $\hat{p}(v) = p(v)$ (i.e., the source terminal is the parent), and let $N_v^0 = N_{\hat{p}(v)}$ otherwise (if the source terminal is not the parent).

If the requirement of v is $\leq |N_v^0|$, then we "copy" the first r_v nodes of N_v^0 to v, i.e., for each of the first r_v nodes in the ordered set N_v^0, we add an edge from v to that node. Step (5) for v is finished after this.

If the requirement of v is $> |N_v^0|$, then we "copy" all the nodes of N_v^0 to v, i.e., for each of the nodes in N_v^0, we add an edge from v to that node. We mark all these new neighbours of v as used w.r.t. v.

Let $\tau \in \{1, \ldots, \ell\}$ be the next available track for v, i.e., v has two neighbours in each of the tracks $Q_1, \ldots, Q_{\tau-1}$, but has only one neighbour in each of the tracks $Q_\tau, \ldots, Q_\ell$.

We repeat the following until step (5) has added a total of r_v neighbours of v. We pick an unused node $x \in \Gamma_v$ with minimum c_{vx}, and mark x as used w.r.t. v. First, suppose that x is free. If the following version of the hop-rule does *not* apply to vx (i.e., there exists no h satisfying the rule), then we insert x into track Q_τ and add the edge vx. Also, we increase τ by one. This causes an increase of $2c_{vx}$ in the cost of the tracks, and an increase of c_{vx} in the cost of the edges from v to the tracks.

To apply the modified hop-rule, we check whether there exists an active terminal $h \neq i = p(v)$ such that

$$c(v,h) \leq (2 + (\beta+1)\alpha)\widetilde{c}(v,x) \leq 14\widetilde{c}(v,x), \qquad \text{and} \qquad \mu_h \leq \widetilde{c}(v,x).$$

If such an h exists, then we add the edge vh_τ and we have $c(v,h_\tau) \leq c(v,h) + \alpha\mu_h \leq (2 + (\beta + 2)\alpha)\widetilde{c}(v,x) \leq 18\widetilde{c}(v,x)$. Also, we increase τ by one.

Now, suppose that x is not free. Then one of the following mutually exclusive cases applies:

(a) $x \in B_h$ for some active terminal h.

(b) $x \in N_h - \bigcup\{B_j \ : \ j \in T^*\}$ for some active terminal h.

(c) x is in one of the tracks, and neither (a) nor (b) applies.

Consider case (a). Note that $h \neq i = p(v)$, because we added edges from v to all nodes in $N_v^0 \supseteq B_i' \supseteq B_i$ (and marked all those nodes as used w.r.t. v) before picking x, hence, $x \notin B_i$. First, suppose that

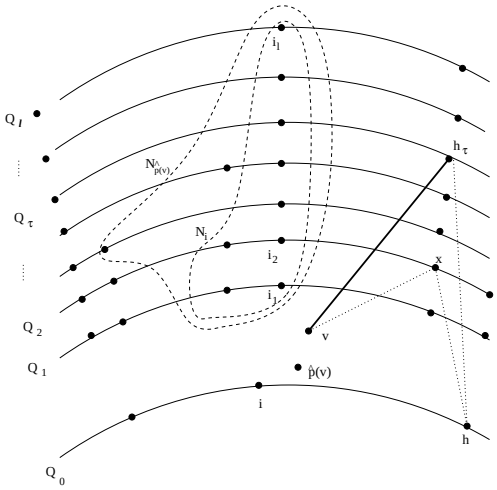

Figure 3: An illustration of step (5)(a) in Section 4, showing v, $p(v) = i$, the prior sibling $\hat{p}(v)$ of v, N_i and $N_{\hat{p}(v)}$; the "dashed edge" vx is replaced by the edge vh_τ since the active terminal h has $x \in B_h$.

$3\widetilde{c}(v,x) \geq \mu_h$. Then we discard x as a neighbour of v and we add the edge (v,h_τ) to H, i.e., the inner neighbour of h in Q_τ is made the outer neighbour of v in Q_τ. Also, we increase τ by one. The new edge has cost $c(v,h_\tau) \leq c(v,x)+c(h,x)+\alpha\mu_h \leq c(v,x)+2\alpha\mu_h \leq (1+6\alpha)\widetilde{c}(v,x) \leq 25\widetilde{c}(v,x)$. Now, suppose that $3\widetilde{c}(v,x) < \mu_h$. Then we have a contradiction because $c(h,i) \leq c(h,x)+c(v,x)+c(v,i) \leq \alpha\mu_h+c(v,x)+\beta\alpha\mu_v+\alpha\mu_i \leq \alpha\mu_h+(1+\beta\alpha)\widetilde{c}(v,x)+\alpha\mu_i = \alpha\mu_h+9\widetilde{c}(v,x)+\alpha\mu_i$ (by $\alpha=4,\beta=2$) $< \alpha\mu_i+(\alpha+3)\mu_h < \alpha\mu_i+\beta\alpha\mu_h$, and moreover, $\mu_i \leq \mu_v \leq \widetilde{c}(v,x) < \mu_h$. To verify the contradiction, recall from step (1) that for active terminals i,h with $\mu_i \leq \mu_h$, we have $c(h,i) \geq \alpha\mu_i+\beta\alpha\mu_h$. See Figure 3 for an illustration.

Consider case (b). As in case (a), note that $h \neq i = p(v)$, because we added edges from v to all nodes in $N_v^0 \supseteq N_i$ (and marked all those nodes as used w.r.t. v) before picking x. First, suppose that $\widetilde{c}(v,x) \geq \widetilde{c}(h,x)$. Then we discard x as a neighbour of v and we add the edge (v,h_τ) to H. Also, we increase τ by one. The new edge has cost $c(v,h_\tau) \leq c(v,x)+c(h,x)+\alpha\mu_h \leq (2+\alpha)\widetilde{c}(v,x) \leq 6\widetilde{c}(v,x)$. Now, suppose that $\widetilde{c}(v,x) < \widetilde{c}(h,x)$. Note that $\widetilde{c}(h,x) = c(h,x) > \alpha\mu_h$, since $x \notin B_h$. Then we have a contradiction by the hop-rule of step (4), because $c(h,i) \leq c(h,x)+c(v,x)+c(v,i) < 2c(h,x)+(\beta+1)\alpha\mu_v < (2+(\beta+1)\alpha)c(h,x) \leq 14c(h,x)$ and $\mu_i \leq \mu_v \leq \widetilde{c}(v,x) < c(h,x)$. Thus the hop-rule of step (4) applies to h and hx, so the active terminal h cannot use the edge hx. Hence, we cannot have $\widetilde{c}(v,x) < \widetilde{c}(h,x)$.

Now, consider case (c). Let w be the first inactive terminal whose processing by step (5) results in the addition of the edge wx (i.e., x changes from a free node to a nonfree node when step (5) processes w). Let $p(w) = h$. Note that $h \neq i$ (i.e., $p(w) \neq p(v)$), otherwise, w is a prior sibling of v (since the edge wx was added during the processing of w by step (5)), and for every prior sibling u of v, each node in N_u is already used w.r.t. v (since we added edges from v to all nodes in $N_v^0 \supseteq N_{\hat{p}(v)}$). First, suppose that $\widetilde{c}(v,x) \geq \widetilde{c}(w,x)$. Then we discard x as a neighbour of v and we add the edge (v,h_τ) to H. Also, we increase τ by one. The new edge has cost $c(v,h_\tau) \leq c(v,x)+c(w,x)+c(h,w)+\alpha\mu_h \leq 2\widetilde{c}(v,x)+\alpha(\beta+2)\mu_w \leq 2\widetilde{c}(v,x)+\alpha(\beta+2)\widetilde{c}(v,x) \leq (2+\alpha(\beta+2))\widetilde{c}(v,x) \leq 18\widetilde{c}(v,x)$. Now, suppose that $\widetilde{c}(v,x) < \widetilde{c}(w,x)$. Then we have a contradiction by the modified hop-rule of step (5), because $c(w,i) \leq c(w,x)+c(v,x)+c(v,i) < 2\widetilde{c}(w,x)+(\beta+1)\alpha\mu_v < (2+(\beta+1)\alpha)\widetilde{c}(w,x) \leq 14\widetilde{c}(w,x)$ and $\mu_i \leq \mu_v \leq \widetilde{c}(v,x) < \widetilde{c}(w,x)$. Thus the modified hop-rule of step (5) applies to w and wx, so the

inactive terminal w cannot use the edge wx. Hence, we cannot have $\widetilde{c}(v,x) < \widetilde{c}(w,x)$.

This completes the description of step (5).

(5′) [SPECIAL HANDLING FOR 1 OR 2 ACTIVE TERMINALS]
Suppose that $n^* = 1$, and $T^* = \{i\}$. Then we ignore the tracks altogether, but we compute the ordered set N_i via step (4) applied to i, and the ordered set N_v for each inactive terminal v by applying step (5) to v. We add the edges from each terminal v (where $v = i$ or v is a child of i) to all the nodes in N_v.

Now, suppose that $n^* = 2$, and $T^* = \{h,i\}$. We proceed as in steps (2)–(5), except that we temporarily allow tracks that consist of exactly two nodes and two copies of the edge between them. In particular, the special track Q_0 consists of nodes h,i and two copies of the edge hi. At the end, for each track consisting of exactly two nodes, we keep only one copy of the edge between them; thus the solution graph H is simple.

Proposition 9 *(i) The total cost of the edges added by step (4) and incident to an active terminal i is $\leq 18\sigma_i$. (ii) At the end of step (4), the total cost of the $\ell+1$ tracks is $\leq (2\ell+2)\,mst(T) + 3\alpha\sigma(T^*)$.*

Proof: For an active terminal i, the total cost of the token arcs (i,h) given to i is $\leq 3\sigma_i$. The cost of the edges added that are incident to i, but excluding the cost due to the token arcs, is $\leq \alpha\sigma_i$ if $|B_i| \geq r_i + 1$ (in this case, $B'_i = B_i$, no token arcs are given to i, and we add edges to the r_i nearest neighbours of i, and then we add $\leq 2\ell - r_i \leq 1.5k - k$ edges of cost $\leq a\mu_i$ to nodes in B_i, for a total cost of $\leq \sigma_i + \frac{k}{2}\alpha\mu_i \leq \alpha\sigma_i$), and otherwise is $\leq (2+(\beta+2)\alpha)\sum_{h\in\Gamma_i} c_{ih} \leq (2+(\beta+2)\alpha)\sigma_i = 18\sigma_i$ (by hop-rule). This proves part (i). For part (ii), observe that the total cost of the $\ell+1$ tracks (that were constructed in step (2) and modified in step (4)) is $\leq (2\ell+2)mst(T) + 3\alpha\sigma(T^*)$; for the second term, note that the contribution of $i \in T^*$ is $\leq 2\sigma_i$ if $|B_i| \leq r_i + 1$, and otherwise is $\leq 2\alpha\mu_i$ for each of $\leq 2\ell = 1.5k$ nodes in B_i, and this sums to $\leq (2\alpha\mu_i)(1.5k) \leq 3\alpha\sigma_i$. ■

Proposition 10 *(i) The total cost of the edges added by step (5) and incident to an inactive terminal v is $\leq 401\sigma_v$. (ii) The total increase in the cost of the tracks in step (5) is at most $2\sigma(T - T^*)$.*

Proof: We claim that the cost of the added edges for an inactive terminal v with parent i is $\leq 401\sigma_v$. Let γ be a constant such that the cost of the added edges incident to an active terminal i is $\leq \gamma\sigma_i$. (From step (4) and Proposition 9, we have $\gamma = 18$.) First, note that if $r_v \leq r_{p(v)}$, then the cost of the added edges incident to v is given by the cost of copying r_v neighbours from the parent $p(v)$ and this cost is $\leq \gamma\sigma_{p(v)} + r_v \cdot c(v,p(v)) \leq 1.5\gamma\sigma_v + r_v \cdot (\beta+1)\alpha\mu_v \leq (1.5\gamma + (\beta+1)\alpha)\sigma_v \leq 39\sigma_v$. Now, assume that $r_v \geq r_{p(v)}$, hence, $\sigma_v \geq \sigma_{p(v)}$.

First, consider the cost incurred in copying the neighbours of the source terminal $\hat{p}(v)$. This cost consists of two components, (i) the cost of copying $r_{p(v)} \leq r_v$ neighbours from the parent $p(v)$, and (ii) the cost of copying the remaining (at most $r_{\hat{p}(v)} - r_{p(v)} \leq k/2$) nodes from $N_{\hat{p}(v)}$. The component (i) is $\leq \gamma\sigma_{p(v)} + r_v \cdot c(v,p(v)) \leq \gamma\sigma_v + r_v \cdot (\beta+1)\alpha\mu_v \leq (\gamma + (\beta+1)\alpha)\sigma_v \leq 30\sigma_v$.

Now, consider component (ii). We claim that component (ii) is $\leq 337\sigma_v$. Consider any node $y \in N_{\hat{p}(v)} - N_{p(v)}$. Let w be the first (earliest processed) sibling of v that has an edge wy (i.e., step (5) added the edge wy while processing w and no prior sibling u of w has $y \in N_u$); possibly, $w \neq \hat{p}(v)$. Call w the *sponsor* of y. By examining the details of step (5), it can be seen that for each node $x \in N_w - N_{\hat{p}(w)}$, there exists a distinct node $x' \in \Gamma_w$ such that $c(w,x) \leq 25\widetilde{c}(w,x')$. Thus for each node $y \in N_{\hat{p}(v)} - N_{p(v)}$, the sponsor w of y has a distinct node $y' \in \Gamma_w$ such that $c(w,y) \leq 25\widetilde{c}(w,y')$. Moreover, there is a distinct node $x' \in \Gamma_v$ such

that $c(w,y') \leq 24\mu_v + c(v,x')$. To see this, first note that $c(v,w) \leq c(v,p(v)) + c(p(v),w) \leq 2c(v,p(v)) \leq 2(\beta+1)\alpha\mu_v \leq 24\mu_v$; next, focus on the nodes x_j in Γ_v ordered by increasing cost of the edge vx_j, say $x_1, x_2, \ldots, x_{r_v}$; suppose that y' is the sth closest neighbour of w; then note that $c(w,y') \leq 24\mu_v + c(v,x_s)$ because each of the nodes x_j in Γ_v has $c(w,x_j) \leq c(v,w) + c(v,x_j)$, hence, for each of the s nodes x_j, $j = 1,\ldots,s$, we have $c(w,x_j) \leq 24\mu_v + c(v,x_j)$. Moreover, $c(v,y) \leq c(v,w) + c(w,y) \leq 24\mu_v + c(w,y)$. Hence, for each node $y \in N_{\hat{p}(v)} - N_{p(v)}$, there is distinct node $x' \in \Gamma_v$ such that $c(v,y) \leq 24\mu_v + 25(24\mu_v + c(v,x')) \leq 624\mu_v + 25c(v,x')$ (since $\mu_w \leq \mu_v$ and $c(w,y) \leq 25\widetilde{c}(w,y') \leq 25(24\mu_v + c(v,x'))$, where w and y' are as above). Then, summing over all nodes $y \in N_{\hat{p}(v)} - N_{p(v)}$, we see that the total cost of these edges vy is $\leq (|N_{\hat{p}(v)} - N_{p(v)}|)(624\mu_v) + 25\sigma_v \leq (k/2)(624\mu_v) + 25\sigma_v \leq 312\sigma_v + 25\sigma_v \leq 337\sigma_v$.

Finally, consider the total cost of the edges from v to the nodes in $N_v - N_{\hat{p}(v)}$. As mentioned above, for each node $y \in N_v - N_{\hat{p}(v)}$, there exists a distinct node $y' \in \Gamma_v$ such that $c(v,y) \leq 25\widetilde{c}(v,y')$. Also, $|N_v - N_{\hat{p}(v)}| \leq r_v - k$, and for each node $y' \in \Gamma_v$ we have $\widetilde{c}(v,y') \leq \mu_v + c(v,y')$. Hence, $\sum\{c(v,y) \ : \ y \in N_v - N_{\hat{p}(v)}\} \leq (r_v - k) \cdot 25\mu_v \ + \ 25\sigma_v \leq 50\sigma_v - 25k\mu_v \leq 50\sigma_v - 25(\frac{2r_v}{3})\mu_v = (50 - 25(\frac{2}{3}))\sigma_v = \frac{100}{3}\sigma_v$.

Summing the three contributions (from components (i), (ii), and the previous paragraph), we see that the total cost of the edges added (by step (5)) incident to an inactive terminal v is $\leq (30 + 337 + 34)\sigma_v \leq 401\sigma_v$.

The total increase in the cost of the tracks in step (5) is at most $2\sigma(T - T^*)$, because during the processing of an inactive terminal v, step (5) may insert each node $x \in \Gamma_v$ into the tracks at an incremental cost of $2c(v,x)$. This completes the proof of the proposition. ■

Remarks: The constant factor of 401 in the above proposition is not optimal. An easy way to improve on it is to replace the constant 25 in the analysis of component (ii) by 19, by tightening the analysis of case (a) in step (5); using the notation from there, recall that this analysis shows that $3\widetilde{c}(v,x) \geq \mu_h$, and hence the newly added edge has cost $\leq 25\widetilde{c}(v,x)$. It can be seen that the '3' may be replaced by $(2\alpha+1)/\alpha = \frac{9}{4}$, and thus the newly added edge has cost $\leq 19\widetilde{c}(v,x)$. We did not optimise the analysis, to avoid further complications.

Proof of Theorem 8: We claim that the cost of the solution subgraph H is $c(H) \leq 900\text{OPT} = O(1)\text{OPT}$. By Propositions 9 and 10 and using $k \geq 4$, we have $c(H) \leq (3\alpha\sigma(T^*) + 2\sigma(T - T^*) + (2\ell+2)mst(T)) + (18\sigma(T^*) + 401\sigma(T - T^*)) \leq 403\sigma(T) + (4)(\frac{k}{2})mst(T) \leq 900\text{OPT}$.

We claim that the solution subgraph H satisfies the connectivity requirements. Consider any pair of inactive terminals s,t. (The proof is similar but simpler for a pair of active terminals, or for one active and one inactive terminal.) First assume that there are at least three active terminals (that is, $|T^*| \geq 3$). Without loss of generality let $r_s = \min(r_s, r_t)$. We claim that H has r_s openly disjoint s,t-paths. Recall that each inactive terminal v has inner neighbours on all $\ell+1$ tracks, and has outer neighbours on the first $r_v - (\ell+1)$ tracks among $Q_1, \ldots, Q_\ell$ (an active terminal v has at least $r_v - \ell$ tracks that have outer neighbours). It follows that we have have $\ell + 1 + r_s - (\ell+1) = r_s$ openly disjoint s,t-paths using these tracks. (One of these s,t-paths consists of a path of the special track $Q_0 = Q$ and the edges $sp(s)$ and $tp(t)$.)

Clearly, the connectivity requirements hold for the case of $|T^*| = 1$. Now, suppose that $|T^*| = 2$. The above arguments still apply unless both s and t have inner and outer neighbours on a track that consists of exactly two nodes, call them x and y. In this case, our track consists of a single edge xy (since we discarded the second copy of xy in step (5′)). Still, this track gives two openly disjoint s,t-paths, namely, s,x,t and s,y,t. Thus it can be seen that the connectivity requirements hold. This completes the proof of Theorem 8. ■

5 The algorithm for node-connectivity SNDP

This section presents a proof of Theorem 4, based on (the algorithms in) Theorems 1, 3. For the sake of motivation, let us obtain an $O(\ln r_{max})$-approximation algorithm for a restricted version of NC-SNDP where every terminal has a requirement r_i and every pair of terminals i, j has the requirement $r_{i,j} = \min(r_i, r_j)$. The method is similar to the method for proving Theorem 3 from Theorems 1 and 8.

Let OPT denote the optimal value of the instance (of restricted NC-SNDP). First, for each $\rho = 1, 2, \ldots, 7$, we apply the algorithm in Theorem 1 to the following instance $\Pi(\rho)$ of the subset ρ-connectivity problem to obtain a solution subgraph $H(\rho)$: we take the requirement of a terminal i in $\Pi(\rho)$ to be $r'_i = 0$ if $r_i < \rho$, and we take $r'_i = \rho$ if $r_i \geq \rho$; the rest of the instance stays the same. By Theorem 1, the cost of $H(\rho)$ is $O(1) \cdot \text{OPT}$. After this, we repeatedly apply the algorithm in Theorem 8 to solve an instance (specified below) of subset $[\rho,\ 1.5\rho]$-connectivity, where ρ is an integer multiple of 4 ($\rho = 8, 12, 16, 24, \ldots$, details later), to obtain a solution subgraph $H'(\rho)$. The instances of subset $[\rho,\ 1.5\rho]$-connectivity are as follows: we take the requirement of a terminal i to be $r'_i = 0$ if $r_i < \rho$, we take $r'_i = r_i$ if $\rho \leq r_i \leq 1.5\rho$, and we take $r'_i = 1.5\rho$ if $r_i > 1.5\rho$. By Theorem 8, the cost of $H'(\rho)$ is $O(1) \cdot \text{OPT}$. We start with $\rho = 8$, and we iterate until $r_{max} \leq 1.5\rho$; after each iteration, we update ρ to the largest integer multiple of 4 that is ≤ 1.5 times the previous ρ. Clearly, the number of iterations is $O(\ln r_{max})$. Finally, we output the solution subgraph H^* for the instance (of restricted NC-SNDP); H^* is the union of all the solution subgraphs $H(\rho)$, $\rho = 1, \ldots, 7$, and $H'(\rho)$, $\rho = 8, 12, \ldots$. Thus H^* is the union of $O(\ln r_{max})$ subgraphs such that each of these subgraphs has cost $O(1) \cdot \text{OPT}$, and so H^* has cost $O(\ln r_{max}) \cdot \text{OPT}$. To see that H^* satisfies the connectivity requirements, note that for every pair of terminals i, j, one of the subgraphs forming H^* has $\min(r_i, r_j)$ openly disjoint i, j-paths, namely, the subgraph $H(\min(r_i, r_j))$ if $\min(r_i, r_j) \leq 7$, otherwise, any subgraph $H'(\rho)$ where ρ satisfies $\rho \leq \min(r_i, r_j) \leq 1.5\rho$.

Our algorithm for metric-cost NC-SNDP is similar to the algorithm described above for the restricted version of NC-SNDP. Let Π^* be an instance of NC-SNDP, and let OPT denote its optimal value. We use k^f to denote an integer multiple of 4 such that $r_{max} \leq 1.5k^f$. We repeatedly apply the algorithm of Theorem 1 (for subset k-connectivity) for $k = 1, \ldots, 7$, and derived instances $\Pi(1), \ldots, \Pi(7)$ to obtain solution subgraphs $H(1), \ldots, H(7)$. Then we repeatedly apply the algorithm of Theorem 8 (for subset $[k,\ 1.5k]$-connectivity) for $k = 8, 12, 16, 24, \ldots, k^f$ and derived instances $\Pi'(8), \Pi'(12), \ldots, \Pi'(k^f)$ to obtain solution subgraphs $H'(8), \ldots, H'(k^f)$. We start these iterations with $k = 8$, and we iterate until $k = k^f$; after each iteration, we update k to the largest integer multiple of 4 that is ≤ 1.5 times the previous k. The construction of the derived instances $\Pi(\rho)$ and $\Pi'(k)$ is described below.

Finally, we output the solution subgraph H^* for Π^*; H^* is the union of all the solution subgraphs $H(k)$, $k = 1, \ldots, 7$, and $H'(k)$, $k = 8, 12, \ldots, k^f$; we call these solution subgraphs the *constituent subgraphs* of H^*. Below, we prove that the cost of each of the constituent subgraphs is at most $O(1) \cdot \text{OPT}$. Clearly, the number of iterations is $O(\ln r_{max})$. Thus H^* is the union of $O(\ln r_{max})$ subgraphs such that each of these subgraphs has cost $O(1) \cdot \text{OPT}$, and so H^* has cost $O(\ln r_{max}) \cdot \text{OPT}$. Below, we prove that H^* satisfies the connectivity requirements, because for every pair of terminals i, j, one of the constituent subgraphs of H^* has $\geq r_{i,j}$ openly disjoint i, j-paths.

We define the derived instances via a well-studied problem in network design, namely, the *generalized Steiner tree* problem, which is as follows: we are given a graph $G = (V, E)$, edge costs c, and $\hat{q}$ sets of terminal nodes $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{\hat{q}}$; the goal is to compute an (approximately) minimum-cost forest F of G such that each terminal set $\hat{D}_m, m = 1, \ldots, \hat{q}$, is contained in a (connected) component of F. Goemans and Williamson [17], based on earlier work by Agrawal et al. [1], gave 2-approximation algorithms for this problem based on the primal-dual method.

Here is the construction for one of the derived instances $\Pi'(k)$; recall that this is an instance of the subset $[k, 1.5k]$-connectivity problem, where k is a fixed parameter. We start from Π^* and construct a requirements graph R with node set T and edge set $E(R)$ as follows. For each terminal pair i, j with $k \leq r_{i,j} \leq 1.5k$ (i.e., the requirement for the pair is within the valid range for our derived instance), we add the edge ij to R. Denote the node sets of the (connected) components of R by $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{\hat{q}}$. Next, we define an instance $\Pi(gst)$ of the *generalized Steiner tree* problem on the graph G with edge costs c (here, G, c are as in Π^*), and with terminal sets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{\hat{q}}$. We solve this auxiliary problem $\Pi(gst)$ by applying the primal-dual algorithm of Goemans and Williamson [17]. Let $F \subseteq E(G)$ be the forest computed by the Goemans-Williamson algorithm, and let $F_1, F_2, \ldots, F_q$ denote the partition of F into connected components. Let the set of terminals in the component of F_m be denoted by D_m, $m = 1, \ldots, q$; thus each set D_m is the union of one or more of the terminal sets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{\hat{q}}$. For each $m = 1, \ldots, q$, we define an instance $\Pi'_m(k)$ of the subset $[k,\ 1.5k]$-connectivity problem as follows: the graph G and the edge costs c are as in Π^*; the set of terminal nodes is D_m, and the requirement r'_i of a terminal $i \in D_m$ is defined to be $\max(r_{i,j} \ : \ \{i,j\} \in E(R))$; clearly, $k \leq r'_i \leq 1.5k$, $\forall i \in D_m$. We take the derived instance $\Pi'(k)$ to be the disjoint union of these instances $\Pi'_m(k)$, $m = 1, \ldots, q$, i.e., we assume that each instance $\Pi'_m(k)$ has its own copy of G and c. To solve $\Pi'(k)$, we take each $m = 1, \ldots, q$, and apply the algorithm in Theorem 8 separately to $\Pi'_m(k)$ to obtain a solution subgraph, call it $H'_m(k)$. (These instances $\Pi'_m(k)$ are pairwise disjoint, and we solve them separately, one by one.) Then we take the union of the subgraphs $H'_1(k), \ldots, H'_m(k)$ and call it $H'(k)$; this is the solution subgraph of $\Pi'(k)$. The cost of the subgraphs $H'_m(k)$, $m = 1, \ldots, q$, is analysed below.

Our reasons for using the auxiliary problem $\Pi(gst)$ for defining the instance $\Pi'(k)$ may be seen from the following example. Suppose that k is large (say $k = \sqrt{n}$) and the edges in $E(R)$ form a matching say $\{\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_{\hat{q}}, t_{\hat{q}}\}\}$, say $\hat{q} = \Theta(n)$. Moreover, suppose that G has a cut $\delta(S)$ such that each edge in this cut is expensive, and some of the edges in $E(R)$ have both ends in S and the remaining edges in $E(R)$ have both ends in $V - S$. Say the optimal solution consists of two disjoint subgraphs, one contained in the subgraph induced by S and the other contained in the subgraph induced by $V - S$. Then we cannot take $\Pi'(k)$ to be a single instance with terminal set $\{s_1, \ldots, s_{\hat{q}}, t_1, \ldots, t_{\hat{q}}\}$, because then every solution subgraph will have $\geq k$ edges from the expensive cut $\delta(S)$. Also, we cannot take $\Pi'(k)$ to consist of $\hat{q}$ separate sub-instances with one sub-instance for each connected component of $R = (T, E(R))$, because the optimal values of these sub-instances may sum to $\hat{q} \cdot \text{OPT}$, and the solution subgraph computed by our algorithm may have cost as high as this (assuming that the algorithm returns the union of the solution subgraphs of these $\hat{q}$ sub-instances). We get around this difficulty by using the Goemans-Williamson algorithm to merge the connected components of $R = (T, E(R))$ into appropriate "clusters" and then we construct a separate sub-instance for each of these "clusters" (these are the sub-instances that we called $\Pi'_1(k), \ldots, \Pi'_q(k)$). The key point is that (i) these sub-instances have pairwise disjoint terminal sets $D_1, \ldots, D_q$, hence, the sum of the $\sigma()$ lower-bounds (used in Theorem 8), namely, $\sum_{m=1}^{q} \sigma(D_m)$, is $\leq$ the $\sigma()$ lower-bound of Π^*, and (ii) the following proof (which is based on the 2-approximation guarantee of Goemans and Williamson) shows that the sum of the $mst()$ lower-bounds for these sub-instances, namely, $\sum_{m=1}^{q} mst(D_m)$, is $\leq O(1)$ times the $mst()$ lower-bound of Π^*. Also, for each sub-instance, the solution subgraph has cost within an $O(1)$ factor of the sum of its $\sigma()$ and $mst()$ lower-bounds. Hence, the union of the solution subgraphs of these sub-instances has cost within an $O(1)$ factor of the optimal value of Π^*.

The construction of the instances $\Pi(\rho)$, $\rho = 1, \ldots, 7$, is similar to that of the instances $\Pi'(k)$. We start with $R = (T, E(R))$ where $E(R)$ consists of terminal pairs $\{i, j\}$ with $r_{i,j} = \rho$. Then we obtain a family of pairwise disjoint sub-instances $\Pi_1(\rho), \Pi_2(\rho), \ldots$ and these sub-instances together form $\Pi(\rho)$.

Proof of Theorem 4: Recall that Π^* denotes the instance of NC-SNDP, OPT denotes the optimal value

of Π^*, and H^* denotes the solution subgraph of Π^* found by our algorithm. The goal is to analyze the cost of the constituent subgraphs of H^* and show that each has cost $\leq O(1) \cdot \text{OPT}$, and then to show that H^* satisfies the connectivity requirements. The proof is based on the following LP (linear programming) relaxation P^* of Π^* that interprets each requirement $r_{i,j}$ as a requirement for $r_{i,j}$ edge-disjoint i,j paths. Thus the optimal value of P^* gives a lower bound on OPT. The LP has a variable x_e, $0 \leq x_e \leq 1$, for each edge $e \in E$; the intention is that each feasible solution H of Π^* gives a zero-one vector $x \in \Re^E$ that satisfies two conditions: $x_e = 1$ iff $e \in H$, and x satisfies the constraints of the LP (though feasible zero-one solutions of the LP may not give feasible solutions of Π^*).

$$\begin{array}{rrcl} P^*: & z^* & = & \min \sum_{e \in E} c_e x_e \\ & \text{subject to} & & \\ & x(\delta(S)) & \geq & \max\{r_{i,j} \ : \ i \in S, j \notin S\}, \quad \forall S \subseteq V \\ & x_e & \geq & 0, \quad \forall e \in E \end{array}$$

Focus on one of the derived instances $\Pi'(k)$ and its associated generalized Steiner tree instance $\Pi(gst)$. We use the notation from the construction of $\Pi'(k)$ given above. Goemans and Williamson [17] proved that the cost of the forest computed by their algorithm is ≤ 2 times the optimal value $z(gst)$ of the following LP relaxation $P(gst)$ of $\Pi(gst)$. The LP has a variable x_e, $0 \leq x_e \leq 1$, for each edge $e \in E$; the intention is that each feasible solution F of $\Pi(gst)$ corresponds to a zero-one vector $x \in \Re^E$ that satisfies two conditions: $x_e = 1$ iff $e \in F$, and x satisfies the constraints of the LP.

$$\begin{array}{rrcl} P(gst): & z(gst) & = & \min \sum_{e \in E} c_e x_e \\ & \text{subject to} & & \\ & x(\delta(S)) & \geq & 1, \quad \forall S \subseteq V : \exists m = 1, \ldots, \hat{q} : \emptyset \neq S \cap \hat{D}_m \neq \hat{D}_m \\ & x_e & \geq & 0, \quad \forall e \in E \end{array}$$

A key observation is that $k \cdot z(gst) \leq \text{OPT}$. To see this, note that multiplying the right-hand-side of any constraint of the LP $P(gst)$ by k gives a constraint that is valid for the LP P^*. (This follows because whenever we have a constraint $x(\delta(S)) \geq 1$ in the LP $P(gst)$, then the node set S separates two terminals v, w such that the requirements graph R has an v,w-path consisting of terminal-pairs $\{i,j\}$ such that $r_{i,j} \geq k$; since the v,w-path of R "crosses" S, one of the terminal-pairs $\{i,j\}$ in the v,w-path "crosses" S, therefore, $\max\{r_{i,j} \ : \ i \in S, j \notin S\} \geq k$, hence, the constraint "$x(\delta(S)) \geq k$" is a valid constraint for the LP P^*.) Consequently, for every feasible solution x^* of the LP P^*, we see that $\frac{1}{k}x^*$ is a feasible solution of the LP $P(gst)$. Moreover, if x^* is an optimal solution of the LP P^*, then we have $z(gst) \leq \frac{1}{k}c(x^*) = \frac{1}{k}z^* \leq \frac{1}{k}\text{OPT}$, or equivalently, $k \cdot z(gst) \leq \text{OPT}$.

Focus on the cost of the solution subgraph $H'(k) = H'_1(k) \cup H'_2(k) \cup \cdots \cup H'_q(k)$, and note that for each $m = 1, \ldots, q$ the cost of $H'_m(k)$ is $O(k) \cdot mst(D_m) + O(1) \cdot \sigma(D_m)$ (by Theorem 8), where D_m denotes the terminal set of $H'_m(k)$. Then the cost of $H'(k)$ is

$$\begin{array}{rcll} O(k) \cdot \sum_{m=1}^{q} mst(D_m) & + & O(1) \cdot \sum_{m=1}^{q} \sigma(D_m) & \\ & \leq & O(k) \cdot \sum_{m=1}^{q} c(F_m) + O(1) \cdot \sigma(T) & (\text{since } mst(D_m) \leq 2c(F_m), \ \forall m = 1, \ldots, q) \\ & \leq & O(k) \cdot c(F) + O(1) \cdot \sigma(T) & \\ & \leq & O(1) \cdot \text{OPT} + O(1) \cdot \sigma(T) & (\text{since } c(F) \leq 2z(gst) \text{ and } z(gst) \leq \text{OPT}/k) \\ & \leq & O(1) \cdot \text{OPT}. & \end{array}$$

A similar analysis for the solution subgraphs $H(1), \ldots, H(7)$ shows that each has cost $\leq O(1) \cdot \text{OPT}$. Thus our claim for the cost of the solution subgraph H^* follows: $c(H^*) = O(\ln r_{max}) \cdot \text{OPT}$.

Finally, let us verify that H^* satisfies the connectivity requirements. Consider any pair of terminals i, j and their requirement $r_{i,j}$. Assume that $r_{i,j} \geq 8$ (otherwise, we are done by a similar but simpler analysis). Focus on an iteration of the algorithm that fixes the parameter k such that $k \leq r_{i,j} \leq 1.5k$. In that iteration, the requirements graph R has the edge $\{i, j\}$, hence, both i, j must be contained in one of the terminal sets $D_1, \ldots, D_q$, say D_1. Now, consider the sub-instance $\Pi'_1(k)$ and its solution subgraph $H'_1(k)$ and note that $H'_1(k)$ must have $\geq r_{i,j}$ openly disjoint i, j-paths because both r'_i and r'_j are $\geq r_{i,j}$ (here, r'_i and r'_j denote the requirements of i and j in $\Pi'_1(k)$) Thus, H^* has $\geq r_{i,j}$ openly disjoint i, j-paths.

This completes the proof of Theorem 4. ■

Acknowledgments. We thank Bill Cunningham, Michel Goemans, Balaji Raghavachari, Ram Ravi, and Santosh Vempala for useful discussions over the years.

Appendix 1: Examples illustrating claims in Section 1

This appendix has details pertaining to Corollary 2 and the remarks following it (in Section 1). In particular, we include a proof of the claim on 2-connected graphs with metric costs, and give examples to show that this claim does not apply to k-connected graphs for $k \geq 3$. Also, we give examples showing that for metric costs, a k-connected graph may be a factor of $\Theta(k)$ times more expensive than a k-edge-connected *multi*-graph. The next result is well known, but we include a proof for the reader's convenience.

Proposition 11 *In a metric graph, a minimum-cost 2-edge connected spanning subgraph has the same cost as a minimum-cost 2-node connected spanning subgraph.*

Proof: Take a counterexample such that the minimum-cost 2-edge connected spanning subgraph H contains as few cut nodes as possible. Clearly H contains at least one cut node v. Let W_1 and W_2 be connected components in $H - \{v\}$. Clearly, v lies on a cycle C_1 in $W_1 \cup \{v\}$ and a cycle C_2 in $W_2 \cup \{v\}$. Let w_1 and w_2 be neighbours of v on C_1 and C_2 respectively. Now, split off the edge-pair vw_1, vw_2, that is add the edge w_1w_2 and remove the edges vw_1 and vw_2. This creates a cycle C on the node set $V(C_1) \cup V(C_2)$. Thus the resulting graph stays 2-edge connected. Note that the number of components in $H - \{v\}$ decreases by one. We repeat this step until $H - \{v\}$ is connected. By the triangle inequality, the cost of the subgraph does not increase. This contradicts our original choice of H. ■

For $k \geq 3$, however, there exist k-edge connected spanning subgraphs of K_n that have lower cost than that of a minimum-cost k-node connected spanning subgraph. To see this let H be the union of two $k+1$ cliques that share exactly one node v. Let the nodes of these cliques be labelled $a_1, a_2, \ldots, a_k, v$ and $b_1, b_2, \ldots, b_k, v$, respectively. Next consider the complete graph K_n on $2k+1$ nodes whose edges costs are given by the shortest-path distances induced by H. That is, every edge in H has cost 1, and every edge in $E(K_n) - E(H)$ has cost exactly 2. Since H itself is k-edge connected we see that K_n contains a k-edge connected spanning subgraph of cost $2\binom{k+1}{2} = k^2 + k$. Now, any k-node connected spanning subgraph of K_n contains at least $\frac{1}{2}(2k+1)k = k^2 + \frac{1}{2}k$ edges. Moreover there must be at least $k-1$ edges of cost 2 between nodes in $a_1, a_2, \ldots, a_k$ and nodes in $b_1, b_2, \ldots, b_k$, otherwise we obtain a node-cut containing less than k nodes. So any k-node connected spanning subgraph of K_n has cost at least $k^2 + \frac{1}{2}k + (k-1)$. This is strictly greater than the cost of the k-edge connected graph H, if $k \geq 3$. The case of $k = 3$ is shown in Figure 4.

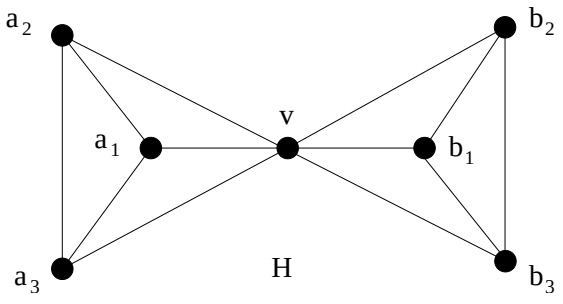

Figure 4: A metric-cost 3-edge connected graph that is strictly cheaper than any 3-node connected (spanning) graph. The edges in H have cost 1, and the edges in $E(K_n) - E(H)$ have cost 2.

Clearly, if the edge costs do not satisfy the triangle inequalities, then the minimum cost of a k-node connected spanning subgraph of K_n cannot be bounded in terms of the cost of a k-edge connected spanning

subgraph. To see this take any k-edge connected graph H that is not also k-node connected (e.g., see Figure 4 for $k = 3$). Let every edge in H have cost 1 and every edge in $E(K_n) - E(H)$ have cost L. Since any k-node connected spanning subgraph of K_n has cost $\geq L$, the claim follows by the choice of L.

Corollary 2 and the other results do not extend to multi-graphs. To see this, let k be an even number, $n - 1 \geq k \geq 2$, and let H be obtained from a cycle on n nodes by taking $\frac{1}{2}k$ copies of each edge. See Figure 5. If each edge in H has cost 1 then a minimum-cost k-edge connected multi-graph has cost $\frac{1}{2}nk$.

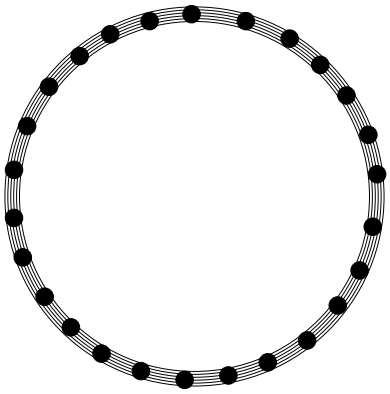

Figure 5: A metric-cost k-edge connected multi-graph that is a factor of $\Theta(k)$ cheaper than any k-node connected spanning subgraph. The edge costs are given by the shortest-paths distances in the cycle.

Let the cost of the other edges of K_n be given by the shortest-path distances in H. Each node has at least k different neighbours in a k-node connected spanning subgraph, so the cost of the edges incident to any node is $\geq 2\sum_{i=1}^{\frac{k}{2}} i = k(\frac{k}{2} + 1)$. Hence, the minimum cost of a k-node connected spanning subgraph is $\geq \frac{1}{4}nk^2$. This is a factor of $\Theta(k)$ times the cost of the k-edge connected graph H.

Appendix 2: Table of Notation & Symbols for Section 4

node set	$V \quad (\|V\| = n)$
set of terminal nodes	$T \quad (\|T\| = n')$
set of active terminal nodes	$T^* \quad (\|T^*\| = n^*)$
terminal nodes (usually active)	h, i, j
inactive terminal nodes	u, v, w
arbitrary nodes (terminals/nonterminals)	x, y
requirement of terminal i	r_i
requirement of terminal pair i, j	$r_{i,j}$
connectivity parameter	$k \quad (k = 0 \pmod 4$ in Section 4)
edge incident to nodes x, y	xy
cost of edge xy	c_{xy} or $c(x, y)$
set of r_i nearest neighbours of i	Γ_i
total cost of edges from i to nodes in Γ_i	σ_i
average cost of an edge from i to nodes in Γ_i	μ_i
normalized cost of edge ix	$\widetilde{c}(i, x) := \max(c_{ix},\ \mu_i)$ (or $\widetilde{c}_{ix}$)
parameters of algorithm in Section 4	$\alpha, \beta, \gamma \quad (\alpha = 4, \beta = 2)$
set of nodes within ball of radius $\alpha\mu_i$ centered at i	B_i
number of tracks	$\ell \quad (\ell = 3k/4$ in Section 4)
tracks	$Q_0, Q_1, Q_2, \ldots, Q_\ell$
index of current track	τ
inner neighbours of active terminal i	$i_1, i_2, \ldots, i_\ell$
parent of inactive terminal v	$p(v)$
ordered set of nodes attaching terminal i to tracks	N_i
cost of MST of subgraph induced by node set X	mst(X)

References

[1] A.Agrawal, P.Klein and R.Ravi, "When trees collide : An approximation algorithm for the generalized Steiner problem on networks," *SIAM Journal on Computing*, **24**, 445–456, 1995. Preliminary version in *Proc. ACM STOC*, 1991.

[2] S.Arora, "Polynomial-time approximation schemes for Euclidean TSP and other geometric problems," *Journal of the ACM*, **45**, 753–782, 1998.

[3] M.Bern and P.Plassmann, "The Steiner problem with edge lengths 1 and 2," *Information Processing Letters*, **32**, 171–176, 1989.

[4] D.Bienstock, E.Brickell and C.Monma, "On the structure of minimum-weight k-connected spanning networks," *SIAM J. Discrete Math.*, **3**, 320–329, 1990.

[5] J.Cheriyan, T.Jordán and Z.Nutov, "On rooted node-connectivity problems," *Algorithmica*, **30**, 353–375, 2001.

[6] J.Cheriyan, S.Vempala and A.Vetta, "Approximation algorithms for minimum-cost k-vertex connected subgraphs," *Proc. 34th ACM STOC*, New York, 306–312, 2002.

[7] J.Cheriyan, S.Vempala and A.Vetta, "An approximation algorithm for the minimum-cost k-vertex connected subgraph," *SIAM Journal on Computing*, **32**, 1050–1055, 2003.

[8] N.Christofides, "Worst case analysis of a new heuristic for the traveling salesman problem," Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, 1976.

[9] A.Czumaj and A.Lingas, "A polynomial time approximation scheme for Euclidean minimum cost k-connectivity," *Proc. 25th ICALP*, LNCS **1443**, 682–694, 1998.

[10] A.Czumaj, A.Lingas and H.Zhao, "Polynomial-time approximation schemes for the Euclidean survivable network design problem," *Proc. 29th ICALP*, LNCS **2380**, 973–984, 2002.

[11] A.Czumaj, M.Grigni, P.Sissokho and H.Zhao, "Approximation schemes for minimum 2-edge-connected and biconnected subgraphs in planar graphs," *Proc. 15th ACM-SIAM SODA*, 489–498, 2004.

[12] G.B.Dantzig, L.R.Ford and D.R.Fulkerson, "Solution of a large-scale traveling-salesman problem," *Operations Research* **2**, 393–410, 1954.

[13] A.Frank, "Connectivity augmentation problems in network design," in *Mathematical Programming: State of the Art 1994*, (Eds. J. R. Birge and K. G. Murty), The University of Michigan, Ann Arbor, MI, 1994, 34–63.

[14] G.L.Frederickson and J.Ja'Ja', "On the relationship between the biconnectivity augmentation and traveling salesman problems," *Theor. Comp. Sci.* **19**, 189–201, 1982.

[15] M.Goemans and D.J.Bertsimas, "Survivable networks, linear programming relaxations and the parsimonious property," *Mathematical Programming*, **60**, 145–166, 1993.

[16] M.Goemans, A.Goldberg, S.Plotkin, D.Shmoys, E.Tardos and D.Williamson, "Improved approximation algorithms for network design problems," *Proc. 5th Ann. ACM-SIAM Symposium on Discrete Algorithms*, 223–232, 1994.

[17] M.Goemans and D.Williamson, "A general approximation technique for constrained forest problems," *SIAM Journal on Computing*, **24**, 296–317, 1995.

[18] K.Jain, "A factor 2 approximation algorithm for the generalized Steiner network problem," *Combinatorica*, **21**(1), 39–60, 2001. Preliminary version in *Proc. 39th IEEE FOCS*, 1998.

[19] S.Khuller, "Approximation algorithms for finding highly connected subgraphs," in *Approximation algorithms for NP-hard problems*, Ed. D.S.Hochbaum, PWS publishing co., Boston, 1996.

[20] S.Khuller and B.Raghavachari, "Improved approximation algorithms for uniform connectivity problems," *Journal of Algorithms* **21**, 434–450, 1996.

[21] G.Kortsarz, R.Krauthgamer and J.R.Lee, "Hardness of approximation for vertex-connectivity network design problems," *SIAM J. Computing* **33**, 704–720, 2004.

[22] G.Kortsarz and Z.Nutov, "Approximating k-node connected subgraphs via critical graphs," *Proc. 36th ACM STOC*, June 2004.

[23] G.Kortsarz and Z.Nutov, "Approximating node connectivity problems via set covers," *Algorithmica* **37**, 75–92, 2003. Preliminary version in *APPROX, Approximation algorithms for combinatorial optimization*, Springer, LNCS **1913**, 194–205, 2000.

[24] L.Lovász, *Combinatorial Problems and Exercises*, North-Holland, Amsterdam, and Akadémiai Kiadó, Budapest, 1979.

[25] C.L.Monma, B.S.Munson and W.R.Pulleyblank, "Minimum-weight two-connected spanning networks," *Mathematical Programming* **46**, 153–171, 1990.

[26] C.L.Monma and D.F.Shallcross, "Methods for designing communication networks with certain two-connectivity survivability constraints," *Operations Research* **37**, 531–541, 1989.

[27] J.S.B.Mitchell, "Guillotine subdivisions approximate polygonal subdivisions: a simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems," *SIAM J. Computing* **28**, 1298–1309, 1999.

[28] M.Stoer, *Design of Survivable Networks*, *Lecture Notes in Mathematics* **1531**, Springer-Verlag, Berlin, 1992.

[29] L.Trevisan, "When Hamming meets Euclid: the approximability of geometric TSP and MST," *SIAM Journal on Computing*, **30**, 475–485, 2001.

[30] V.V.Vazirani, *Approximation Algorithms*, Springer-Verlag, Berlin, 2001.

[31] D.Williamson, M.Goemans, M.Mihail and V.Vazirani, "A primal-dual approximation algorithm for generalized Steiner network problems," *Combinatorica* **15**, 435–454, 1995.