An Approximation Algorithm for Minimum-Cost Network Design

Yishay Mansour * David Peleg †

December 29, 1998

Abstract

This paper considers the problem of designing a minimum cost network meeting a given set of traffic requirements between n sites, using one type of channels of a given capacity, with varying set-up costs for different vertex pairs (comprised of a fixed part plus a part dependent on the pair). An approximation algorithm is proposed for this problem, guaranteeing a solution whose cost is greater than the optimum by a factor of at most $O(\log n)$ (or constant in the Euclidean case). The algorithm is based on the use of a light-weight distance-preserving spanner.

1 Introduction

The network design problem has been intensively studied at least since the 70's [KC74, BF77, GFCE74]. A number of heuristics have been developed for it, and various solutions were given for specific cases of the problem, or for specific subproblems (see [K93] and the references therein). The goal of the current note is to give a simple approximation algorithm for the general problem.

The *minimum cost network design (MCND)* problem can be formalized as follows. There are n sites, forming the vertex set of the network, $V = \{1, \ldots, n\}$. The traffic requirements among vertex pairs are specified by an $n \times n$ *requirement* matrix $\mathcal{R} = (r_{i,j})$, where $r_{i,j}$ is the amount of traffic required to be transmitted between vertices i and j. We are required to design a network connecting these sites, by specifying which vertex pairs are to be connected by links, and what capacity should each of these links possess. These capacities should be large enough to enable the network to handle all the traffic requirements that exist in the network.

Notice that a solution to the problem can be completely characterized by specifying, for each pair of vertices i and j, the routes along which the traffic $r_{i,j}$ will be transmitted. Our model allows traffic to be split along several routes, but our solution does not take advantage of this assumption, and specifies a single route $\rho_{i,j}$ for each vertex pair i, j. The collection of routes for all vertex pairs is denoted $\hat{\rho}$. Fixing the routes determines exactly how much traffic will be transmitted over each edge of the network. Specifically, given a route collection $\hat{\rho}$ and an edge $e = (i,j)$, let $Q(e)$ denote the collection of vertex pairs whose route goes through e, $Q(e) = \{(i,j) \mid e \text{ occurs on } \rho_{i,j}\}$, and define the *load* induced by $\hat{\rho}$ on e as $q_e = \sum_{(i,j)\in Q(e)} r_{i,j}$. Clearly, in order for our link assignment to be feasible, each edge e must have at least q_e capacity.

In practice, the network designer can choose a link capacity from among a pre-fixed, limited set of available bandwidths; intermediate capacities are not available. The model adopted in this paper assumes that capacity is available in whole units, i.e., there is a basic unit of capacity, and the links

*Computer Science Dept., Tel-Aviv Univerity, Tel-Aviv Israel. mansour@cs.tau.ac.il.

†Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, 76100 Israel. peleg@wisdom.weizmann.ac.il. Supported in part by a Walter and Elise Haas Career Development Award and by a grant from the Israel Science Foundation.

available to us have an integral number of capacity units. Hence a feasible solution must assign to each edge e at least $\lceil q_e \rceil$ capacity units.

Our model makes the assumption that the cost associated with placing a link between the vertices i and j is composed of two components. The first is a *fixed initial price* F associated with setting the link, representing the initial set-up cost and the overhead incurred by the endpoints. The second cost component is dependent on the capacity required for the link. This component is specified by an $n \times n$ matrix $\mathcal{P} = (p_{i,j})$, where $p_{i,j}$ is the *price factor* for the link connecting i and j per capacity unit. This factor is typically proportional to the distance between the sites i and j. This is a fair approximation of the pricing of links in practice.

We also make the reasonable assumption that the costs of $\mathcal{P}$ obey the triangle inequality, i.e., it is always cheaper to ship one unit of capacity over a direct edge than over an alternative route composed of a number of hops. (In practice, this will normally be the case, since the direct edge can be built along the cheapest route, even when this route goes physically through other vertices.)

In summary, the total cost incurred by a given choice of route collection $\hat{\rho}$, denoted $C(\hat{\rho})$, is as follows. Let E' denote the set of edges utilized by these routes, i.e., $E' = \{e \mid Q(e) \neq \emptyset\}$. Then

$$C(\hat{\rho}) = |E'| \cdot F + \sum_{(i,j) \in E'} p_{i,j} \cdot \lceil q_{(i,j)} \rceil .$$

Our problem is to find a route assignment that minimizes this cost.

While our abstraction captures many essential elements of the network design problem, it should be realized that it does not capture the full complexity of the problem in practice. In particular, issues such as minimum connectivity or network diameter are not handled in our solution.

We show that the MCND problem is NP-hard to approximate within a factor less than $c \log n$, for some constant $c > 0$. Consequently, we focus on finding approximate solutions for the problem. Let C^* denote the optimal possible cost, given the matrices $\mathcal{R}$, $\mathcal{P}$ and the constant F. Given an algorithm A for the problem, let C^A denote the cost of the solution generated by this algorithm. Then A is an approximation algorithm with approximation ratio α if for every input $(\mathcal{R}, \mathcal{P}, F)$, the solution provided by the algorithm satisfies $C^A \leq \alpha \cdot C^*$. We give a polynomial time approximation algorithm for the problem, with approximation ratio $O(\log n)$. This is the best possible, up to a constant, unless P=NP.

For the special case where the network vertices are located in the two-dimensional plane, and the price matrix $\mathcal{P}$ represents Euclidean distances (i.e., $p_{i,j}$ is proportional to the Euclidean distance between i and j), the approximation factor of our algorithm can be improved to a small constant.

Numerous variants of the network design problem were studied in the literature over the last three decades. Some of those variants are rather close to our problem. Among those are the *survivable* or k-connected network problem, cf. [S92] and Chapters 4 and 6 of [H95], and the *maximum multicommodity flow* or *maximum concurrent flow* problems, cf. Chapter 5 of [H95]. In a number of cases, variants of the aforementioned problems have been given logarithmic ratio approximation algorithms. We are unaware, however, of previous results directly applicable to our problem. Moreover, the techniques used for tackling those problems are entirely different from ours.

2 Preliminaries

For the solution, we need the following concepts. Consider a weighted n-vertex graph $G = (V, E, \omega)$, where $\omega : E \mapsto R^+$ is a weight function assigning an arbitrary positive weight to each of the edges. For any subgraph $G' = (V', E', \omega)$ of G, let $\omega(G')$ denote the total weight of G', i.e., $\omega(G') = \sum_{e \in E'} \omega(e)$. The *minimum-weight spanning tree*, or *MST* of G is a spanning tree T_M minimizing $\omega(T_M)$.

For any subgraph $G' = (V', E', \omega)$ of G, we let $dist_{G'}(u,v)$ be the weighted distance from u to v in G', i.e., the minimum of $\omega(p)$ over all paths p from u to v in G'. For a spanning subgraph $G' = (V, E', \omega)$, let $\texttt{Stretch}(G') = \max_{v,u \in V}\{dist_{G'}(u,v)/dist_G(u,v)\}$. The subgraph G' is said to be a κ-spanner for G if $\texttt{Stretch}(G') \le \kappa$.

Spanners have been studied in a number of papers, motivated by applications in diverse contexts such as distributed systems, communication networks, robotics and computational geometry (cf. [PU89, PS89, ADDJS93, ABP91, CDNS92]). The spanners we use for our present purposes are the low-stretch, sparse, light-weight spanners constructed by the simple greedy algorithm of [ADDJS93, CDNS92]. Alternatively, one can employ the spanner construction algorithm of [ABP91]. In case the variance in link prices in the matrix $\mathcal{P}$ is not extremely large (specifically, as long as the ratio between the largest and smallest price is bounded by a polynomial in n), that algorithm yields comparable spanners to those of [ADDJS93], and it has the additional advantage of having an efficient distributed implementation (whereas the algorithm of [ADDJS93] is inherently sequential).

The greedy algorithm of [ADDJS93] is a generalization of Kruskal's algorithm for building a minimum spanning tree. It takes as input a weighted graph $G = (V, E, \omega)$, and a parameter $\kappa > 1$, and constructs a κ-spanner $G' = (V, E')$ as follows. The algorithm first sets $E' \leftarrow \emptyset$, and sorts the edges of E in nondecreasing order of weights, $E = \{e_1, \ldots, e_m\}$. It then examines the edges one by one in that order. For each edge $e_i = (u,v)$, it checks to see whether $dist_{G'}(u,v) \le \kappa \cdot \omega(e_i)$, where $G' = (V, E')$. If so, then it discards e_i. Otherwise, it adds e_i to E'. In the end, the algorithm outputs $G' = (V, E')$.

We rely on the following theorem, implied as a corollary of Theorem 1.1 in [CDNS92]. (The size bound is not mentioned explicitly in [CDNS92], but can be derived from their analysis.)

Theorem 2.1 [CDNS92] *For every n-vertex weighted graph G, the greedy algorithm (with parameter $\kappa = \log n$) constructs a spanner G' for G with the following properties: (1) $\texttt{Stretch}(G') \le \log n$, (2) $|E(G')| = O(n)$, and (3) $\omega(G') = O(\log n) \cdot \omega(MST)$.*

Let us now turn to spanners for the more restricted Euclidean case. In this setting, G is a complete graph whose vertices are points in the plane and $dist(u,v)$ is the distance from the point u to the point v in some L_p metric. The specific result we shall use in what follows is the following.

Theorem 2.2 [ADDJS93] *For every n-vertex Euclidean graph G, there is a polynomial algorithm for constructing a spanner G' for G with the following properties: (1) $\texttt{Stretch}(G') = O(1)$, (2) $|E(G')| = O(n)$, and (3) $\omega(G') = O(1) \cdot \omega(MST)$.*

3 The network design approximation algorithm

Let $V_{\mathcal{R}}$ be the set of *client vertices* of $\mathcal{R}$, namely, the vertices with some nonzero requirements, $V_{\mathcal{R}} = \{i \mid \exists j \; r_{i,j} > 0\}$. Similarly, let $E_{\mathcal{R}} = \{(i,j) \mid r_{i,j} > 0\}$. Let $G_{\mathcal{R}}$ denote the graph of nonzero requirements of $\mathcal{R}$ (namely, $G_{\mathcal{R}} = (V_{\mathcal{R}}, E_{\mathcal{R}})$). Throughout this section we assume that $G_{\mathcal{R}}$ is connected. In the next section we outline how this assumption can be discarded.

We use the greedy spanner algorithm of [ADDJS93] to solve our network design problem as follows.

The network design algorithm

1. Given the price matrix $\mathcal{P}$, construct a $\log n$-spanner G' for the graph $(V, V \times V, \omega)$ based on the weight function $\omega_{i,j} = p_{i,j}$ as in Theorem 2.1.

2. For every vertex pair i, j, set $\rho_{i,j}$ to be some shortest path connecting i and j in the spanner G' (where again, distances are measured with respect to the weight function $\omega = \mathcal{P}$).

As discussed above, specifying the routes completely characterizes the solution, i.e., the required link capacities and their total cost. It is also quite straightforward to verify that the algorithm runs in time polynomial in n. It remains to analyze the quality of the constructed solution, and compare it with the optimal one. For doing so, it is useful to first obtain some bounds on the cost of the optimal solution itself. We will rely on the following two straightforward bounds.

Lemma 3.1 *The cost C^* of the optimal solution for the problem is bounded below as follows:*

(a) $C^* \geq \omega(MST) + F\cdot(n-1)$, where $\omega(MST)$ denotes the weight of the minimum-weight spanning tree for the graph.

(b) $C^* \geq \sum_{1\leq i,j\leq n} r_{i,j}\cdot p_{i,j} + F\cdot(n-1)$.

Proof: The first claim follows from the fact that the network must be connected, since by our assumption $G_{\mathcal{R}}$ is connected. Consequently, it should contain (at least) some spanning tree of the graph. The second claim follows from the fact that even if we manage to satisfy the requirements exactly and waste nothing, we still must pay, for each requirement $r_{i,j}$, at least the minimal price factor (namely $p_{i,j}$, by the triangle inequality). ∎

Theorem 3.2 *The network design algorithm constructs a route collection $\hat{\rho}$ whose cost satisfies $C(\hat{\rho}) \leq O(\log n)\cdot C^*$.*

Proof: Let E' denote the set of spanner edges, and let $\tilde{E}$ denote the set of edges utilized by the routes ($\tilde{E}\subseteq E'$). For every edge $e\in\tilde{E}$, let $\Delta(e) = \lceil q_{(i,j)}\rceil - q_{(i,j)}$. Let $A = \sum_{(i,j)\in\tilde{E}} p_{i,j}\cdot q_{(i,j)}$, and $B = |\tilde{E}|\cdot F + \sum_{(i,j)\in\tilde{E}} p_{i,j}\cdot\Delta_{i,j}$. Then $C(\hat{\rho}) = A + B$.

We shall now bound each of these terms separately with respect to the cost paid by the optimal solution. First, observe that A can be rewritten as $A = \sum_{1\leq i,j\leq n} r_{i,j}\cdot\omega(\rho_{i,j})$. Therefore, by part (1) of Theorem 2.1, we have that $A \leq \sum_{1\leq i,j\leq n} r_{i,j}\cdot\log n\cdot p_{i,j}$. By part (b) of Lemma 3.1 it follows that $A\leq \log n\cdot C^*$.

Next let us bound B. Since $\Delta_{i,j} < 1$ for every i,j, it follows that $B \leq |\tilde{E}|\cdot F + \sum_{(i,j)\in\tilde{E}} p_{i,j}$. By part (2) of Theorem 2.1, $|\tilde{E}| = O(n)$. By part (3) of Theorem 2.1,

$$\sum_{(i,j)\in\tilde{E}} p_{i,j} \leq \sum_{(i,j)\in E'} p_{i,j} = O(\log n)\cdot\omega(MST).$$

Put together, we have that $B \leq O(n)\cdot F + O(\log n)\cdot\omega(MST)$. By part (a) of Lemma 3.1 it follows that $B \leq O(\log n)\cdot C^*$. Combining these two bounds, we get that $C(\hat{\rho}) = A + B \leq O(\log n)\cdot C^*$, establishing the desired bound. ∎

For the Euclidean case, in which the sites are situated on the two-dimensional plane, and the price matrix $\mathcal{P}$ is dependent on the distances, we have a better solution, based on the existence of stronger spanners, as stated in Theorem 2.2. Specifically, the same algorithm as in the general case, only using the special construction for the Euclidean case, yields an improved result of constant approximation ratio.

Theorem 3.3 *In the Euclidean case, the network design algorithm constructs a route collection $\hat{\rho}$ whose cost satisfies $C(\hat{\rho}) \leq O(1)\cdot C^*$.* ∎

4 Discarding the connectivity assumption

The solution given in the previous section relies on the assumption that the graph $G_{\mathcal{R}} = (V, E_{\mathcal{R}})$ induced by the nonzero requirements of $\mathcal{R}$ is connected. Let us now outline the modifications necessary in order to solve the problem without this assumption. Note that it may not suffice to handle each connected component of $G_{\mathcal{R}}$ separately, since an integrated solution, possibly using edges between different connected components, may potentially have lower cost.

Consider the *generalized Steiner tree* problem, introduced in [AKR91]. Given a vertex set $V = \{1, \ldots, n\}$ and a set of pairs $Q \subseteq V \times V$, a *Steiner forest* for Q is a forest H on V with the property that for every pair $(i, j) \in Q$, there exists a tree $T \in H$ containing both i and j. Given also a complete weight matrix ω on V, the generalized Steiner tree problem calls for finding the lightest Steiner forest for Q. This problem is given a 2-approximation in [AKR91].

To approximate our network design problem, we do the following.

The modified algorithm

1. Apply the 2-approximation algorithm of [AKR91] to the instance at hand, taking $\omega = \mathcal{P}$ and $Q = E_{\mathcal{R}}$, and obtain a light Steiner forest $H = \{T_1, \ldots, T_h\}$ for Q.
2. For each tree $T_\ell \in H$, let $V_\ell = V(T_\ell)$ and let $\mathcal{P}_\ell$ be the restriction of $\mathcal{P}$ to V_ℓ.
3. For every tree T_ℓ separately, apply the algorithm of the previous section on the subgraph $G_\ell = (V_\ell, V_\ell \times V_\ell, \mathcal{P}_\ell)$.

Note that for every client vertex i there is a unique ℓ such that all the routes involving i occur inside the spanner generated for G_ℓ. Hence the resulting network provides a feasible solution for the entire problem.

For analyzing the approximation ratio, let us first consider the cost of the optimal solution. Let H^* be the lightest Steiner forest for Q. Let $m = |V_{\mathcal{R}}|$ denote the number of client vertices (for which there is some nonzero requirement in $\mathcal{R}$).

Lemma 4.1 *The cost C^* of the optimal solution for the problem is bounded below as follows:*

(a) $C^* \geq \omega(H^*)$,

(b) $C^* \geq F \cdot m/2$,

(c) $C^* \geq \sum_{1 \leq i,j \leq n} r_{i,j} \cdot p_{i,j}$.

Proof: The first claim follows from the fact that the optimal solution must connect all the pairs of Q, and H^* represents the lightest way of doing so. The second claim holds since the network edges must touch every client vertex of $V_{\mathcal{R}}$, and each edge touches two vertices. The third claim follows as in Lemma 3.1. ■

Theorem 4.2 *The network design algorithm constructs a route collection $\hat{\rho}$ whose cost satisfies $C(\hat{\rho}) \leq O(\log n) \cdot C^*$.*

Proof: For each G_ℓ, let $n_\ell = |V_\ell|$ and let δ_ℓ denote the number of client vertices in G_ℓ, i.e., $\delta_\ell = |V_\ell \cap V_{\mathcal{R}}|$. Also, let E'_ℓ denote the set of edges in the spanner constructed for G_ℓ, and let $\tilde{E}_\ell$ denote the set of edges utilized by the routes over G_ℓ. Let $\tilde{E} = \cup_\ell \tilde{E}_\ell$.

Note that due to the triangle inequality, the tree T_ℓ does not contain any Steiner (non-client) vertex of degree 1 or 2, and therefore $n_\ell \leq 2\delta_\ell$. Hence overall $\sum_\ell n_\ell \leq 2\sum_\ell \delta_\ell = 2m$. It follows by Thm. 2.1 that

$$|\tilde{E}| = \sum_\ell |\tilde{E}_\ell| = \sum_\ell O(n_\ell) = O(m) \ . \tag{1}$$

Note that for each component V_ℓ, T_ℓ is the MST for the subgraph $(V_\ell, V_\ell \times V_\ell, \mathcal{P}_\ell)$. Also, by Thm. 2.1 again, $\omega(\tilde{E}_\ell) = O(\omega(T_\ell))$. Therefore, since H is a 2-approximate solution for the generalized Steiner tree problem, we have

$$\omega(\tilde{E}) = O(\omega(H)) = O(\omega(H^*)). \tag{2}$$

The rest of the proof follows as in the proof of Thm 3.2, analyzing the cost components A and B separately over each subgraph G_ℓ, and relying on the above (1) and (2) to bound A and B. ∎

Similarly, for the Euclidean case we have

Theorem 4.3 *In the Euclidean case, the network design algorithm constructs a route collection $\hat{\rho}$ whose cost satisfies $C(\hat{\rho}) \leq O(1) \cdot C^*$.* ∎

5 Hardness

We conclude by pointing out that unless P=NP, there is no polynomial-time algorithm for approximating the problem with ratio better than $c \log n$ for some constant $c > 0$. This follows from a trivial reduction from the set cover problem, described next.

Given an instance $\langle U, \mathcal{S}\rangle$ of the set cover problem, where $U = \{u_1, \ldots, u_n\}$ is the n-element universe and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is the collection of sets, construct an instance of the network design problem as follows.

Let the vertex set contain $n + m + 1$ vertices $V = \{0, 1, \ldots, n + m\}$. Intuitively, vertices $1, \ldots, n$ represent the elements of U and vertices $n+1, \ldots, n+m$ stand for the sets of $\mathcal{S}$. The traffic requirements are $r_{0,i} = 1/n$ for $1 \leq i \leq n$, and $r_{i,j} = 0$ elsewhere. The initial price is set to $F = 0$, and the capacity price factors are set to be $p_{0,n+j} = 1$ for $1 \leq j \leq m$, $p_{i,n+j} = 0$ for every pair $1 \leq i \leq n$ and $1 \leq j \leq m$ such that $u_i \in S_j$, and $p_{i,j} = \infty$ elsewhere. It is easy to verify that to avoid infinite cost, it is necessary to route traffic from vertex 0 to each vertex $1 \leq i \leq n$ via some vertex $n + 1 \leq j \leq n + m$ such that S_j contains u_i. Hence for any feasible solution $\hat{\rho}$ for this instance of the problem, the cost $C(\hat{\rho})$ is precisely the size of the set $J(\hat{\rho})$ of vertices in the range $n + 1 \leq j \leq n + m$ occurring on some route from 0 to $1 \leq i \leq n$. Moreover, the solution $\hat{\rho}$ is feasible iff the sets corresponding to the vertices of $J(\hat{\rho})$ form a cover for U (whose cost is clearly $C(\hat{\rho})$ as well). Hence an approximation algorithm with ratio α for the network design problem yields an algorithm for the set cover problem with the same ratio.

It follows from [LY94, F96, RS97] that there exists a constant $c < 1$ such that the network design problem admits no $c \cdot \ln n$-ratio approximation, unless $P = NP$.

Acknowledgement

We are grateful to Aaron Kershenbaum for introducing us to the problem and for a number of helpful discussions and suggestions. We also thank Dan Bienstock for his critical remarks, and Tselly Regev and Dror Gutgold for their helpful comments.

References

[AKR91] A. Agarwal, P. Klein and R. Ravi. When trees collide: an approximation algorithm for the generalized Steiner problem in networks. In *Proc. 23rd ACM Symp. on Theory of Computing*, 134–144, 1991.

[ABP91] B. Awerbuch, A. Baratz and D. Peleg. Efficient Broadcast and Light-Weight Spanners. Unpublished Manuscript, Nov. 1991.

[ADDJS93] I. Althöfer, G. Das, D. Dobkin D. Joseph and J. Soares. On sparse spanners of weighted graphs. In *Discrete and Computat. Geometry*, 9:81–100, 1993.

[BF77] R.R. Boorstyn and H. Frank. Large-scale network topological optimization. *IEEE trans. on commun.*, COM-25:29–47, 1977.

[CDNS92] B. Chandra, G. Das, G. Narasimhan and J. Soares. New sparseness results on graph spanners. In 8^{th} *Symp. on Computational Geometry*, 1992.

[F96] U. Feige. A threshold of $\ln n$ for approximating set cover. In *Proc. 28th ACM Symp. on Theory of Computing*, 314-318, 1996.

[GFCE74] M. Gerla, H. Frank, W. Chou and J. Eckl. A cut saturation algorithm for topological design of packet switched communication networks. *NTC 74*, pages 1074–1085, 1974.

[H95] D.S. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Co., Boston, MA, 1995.

[K93] A. Kershenbaum. *Telecommunications network design algorithms.* McGraw-Hill Book Co., 1993.

[KC74] A. Kershenbaum and W. Chou. A unified algorithm for designing multidrop teleprocessing networks. *IEEE trans. on Commun.*, COM-22, 11:1762–1772, 1974.

[LY94] C. Lund and M. Yannakakis. On the hardness of Approximating Minimization Problems. *J. of the ACM* **41**, (1994), 960–981.

[MP94] Y. Mansour and D. Peleg. An Approximation Algorithm for Minimum-Cost Network Design. Technical Report CS94-22, the Weizmann Institute of Science, Rehovot, Israel, 1994.

[PS89] D. Peleg and A.A. Schäffer. Graph spanners. *J. of Graph Theory*, 13:99–116, 1989.

[PU89] D. Peleg and J.D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. on Comput.*, 18(2):740–747, 1989.

[RS97] R. Raz and S. Safra. A sub constant error probability low degree test, and a sub constant error probability PCP characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, 1997, 475–484.

[S92] M. Stoer. *Design of Survivable Networks.* Springer-Verlag, 1992.