Approximation Algorithms for a Capacitated Network Design Problem*

R. Hassin† *R. Ravi*‡ *F. S. Salman*§

December 17, 2001

Abstract

We study a capacitated network design problem with applications in local access network design. Given a network, the problem is to route flow from several sources to a sink and to install capacity on the edges to support the flow at minimum cost. Capacity can be purchased only in multiples of a fixed quantity. All the flow from a source must be routed in a single path to the sink. This NP-hard problem generalizes the Steiner tree problem and also more effectively models the applications traditionally formulated as capacitated tree problems. We present an approximation algorithm with performance ratio $(\rho_{ST} + 2)$ where ρ_{ST} is the performance ratio of any approximation algorithm for minimum Steiner tree problem. When all sources have the same demand value, the ratio improves to $(\rho_{ST} + 1)$ and in particular, to 2 when all nodes in the graph are sources.

Keywords: Network design, approximation algorithms, routing flow, capacity installation

1 Introduction

We consider the problem of capacity installation and the routing of traffic from a set of source nodes to a single sink node in a network. We model applications in which capacity can be purchased in multiples of a fixed quantity. In telecommunication network design this corresponds to installing transmission facilities such as fiber-optic cables on the edges of a centralized network, and in transportation networks this applies to assigning vehicles of fixed capacity to routes from several destinations to a single hub node. A typical telecommunication network consists of a backbone network and several local access networks. The backbone provides very high speed connection between gateway nodes (service hubs) and each local access network collects traffic at a centralized gateway node through which the backbone is accessed. The performance measures for the effectiveness of a network design include cost, reliability and quality of service. While reliability is a major criterion for a backbone network, cost and quality of service may play important roles in the design of local access networks. The problem we study models the topological design of a cost-effective local access network.

*A preliminary version of this paper appeared in the *Proceedings of the APPROX'2000*, LNCS Vol. 1913, 2000.

†Department of Statistics and Operations Research, Tel-Aviv University, Tel Aviv 69978, Israel. This work was done when this author visited GSIA, Carnegie Mellon University. Email: hassin@post.tau.ac.il

‡GSIA, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Supported by an NSF CAREER grant CCR-9625297. Email: ravi@cmu.edu

§Krannert School of Management, Purdue University, West Lafayette, IN 47907, USA. Supported by an IBM Corporate Fellowship while this work was done. Email: salmanf@mgmt.purdue.edu

The *capacitated network design problem*, which we will refer to as CND in short, can be stated as follows. We are given an underlying undirected graph $G = (V, E)$. A subset S of nodes is specified as sources of traffic and a single sink t is specified. Each source node $s_i \in S$ has a positive integer-valued demand dem_i, all of which must be routed to t via a single path, that is flow cannot be bifurcated. Capacity of an edge e of G is provided in multiples of a modularity U, a given positive integer, by purchasing and installing one or more of the transmission facility at cost c_e per each one. The problem is to find a minimum cost installation of facilities that provides sufficient capacity to route all of the demand simultaneously. The problem requires choosing a path from each source to the sink node and finding the number of facilties to be installed on each edge such that when all the demand is routed, the total flow on the edge is at most the capacity. In other words, traffic originating in different sources may share the capacity on the installed cables and the capacity installed on an edge has to be at least as much as the *total* traffic routed through this edge in both directions.

Based on the telecommunication application we refer to the transmission facility installed on the edges as the "cable", even though it could represent other modes of transmission. In telecommunications, typically a set of cables with differing capacities is available to the network designer. Our problem corresponds to the case where a single cable type is used. The CND problem has been studied in the literature as the single-facility network loading problem, together with its generalizations such as the multicommodity and the multiple facility cases. For a survey on exact solution methods in this area, the reader is referred to the chapter on multicommodity capacitated network design by Gendron, Crainic and Frangioni in [SS99], and the chapter on network design by Balakrishnan, Magnanti and Mirchandani in [DMM97]. In spite of the recent computational progress, the size of the instances that can be solved to optimality in reasonable time contains no more than 50 nodes and 100 edges, whereas real-life instances may contain hundreds of nodes and edges.

The capacitated network design problem is NP-hard since it generalizes the Steiner tree problem which has been shown to be NP-hard [GJ77]. Given a graph with a subset of vertices distinguished as the set of terminals, and costs on the edges, the *Steiner tree problem* is to find a minimum-cost subtree spanning the set of terminals. In the CND problem, when cable capacity is larger than the total demand in the network, one copy of the cable will be installed on every edge that carries flow. This special case of our problem is equivalent to a Steiner tree problem, where the set of terminals to be connected is the set of the source nodes and the sink node.

The NP-hardness of CND implies that the existence of an algorithm that finds an optimal solution in polynomial time is very unlikely. Therefore, we focus on obtaining provable near-optimal solutions in polynomial time. For CND, a constant factor approximation was obtained earlier by Salman et al. in [SCR+97]. The algorithm in [SCR+97] is based on an algorithm of Mansour and Peleg [MP94], that approximates the multicommodity and the single cable type problem in an n-node graph with an $O(\log n)$ performance ratio. While Mansour and Peleg used a spanner to route traffic, Salman et al. [SCR+97] showed that routing through a Light Approximate Shortest Path Tree (defined in [KRY93]) gives a worst-case performance ratio of 7 for the single sink problem. When all the nodes in the input network except the sink node are source nodes, the approximation ratio in [SCR+97] reduces to $(2\sqrt{2} + 2)$. Other constant factor approximations for CND also follow from the work of Andrews and Zhang [AZ98] who gave an $O(k^2)$-approximation for the single sink problem with k cable types, and the work of Garg et al. [GKK+01] who gave an improved $O(k)$-approximation for the same problem, but the resulting constant factors are rather high.

In this paper, we present an approximation algorithm with a better approximation ratio by making use of the relation of the CND problem to the Steiner tree problem. The algorithm utilizes a Steiner tree when the demand is low compared to the cable capacity and whenever the demand accumulates to a value close to the cable capacity, the algorithm sends the aggregated demand by a shortest path to the sink. We show that this algorithm has a worst-case ratio $(\rho_{ST}+2)$, where ρ_{ST} is the performance ratio of the Steiner tree approximation. For the case when demand is uniform, that is every source has the same demand value, our approximation ratio improves to $(\rho_{ST}+1)$. When all the nodes in the input network except the sink node are source nodes, we first find a minimum spanning tree instead of a Steiner tree. As a result, the approximation ratio reduces to 3 with non-uniform demands, and to 2, with uniform demands.

Although the Steiner tree problem is NP-hard even with Euclidean or rectilinear costs [GJ77], a Steiner tree with approximately minimum cost can be constructed in polynomial time. A minimum cost tree spanning all the terminals has cost at most twice the cost of an optimal Steiner tree [TM80]. Furthermore, approximation algorithms with improved worst-case ratios have been developed in a series of papers [Z93, BR94, KZ97, PS97, HP99, RZ00] over the last two decades. As a result, for graphs with arbitrary costs, the worst-case ratio was gradually decreased from 2 to 1.55. Robins and Zelikovsky [RZ00] gave the currently best known approximation ratio of 1.55. On the other hand, it has also been shown that the Steiner tree problem can not be approximated within a factor of $1+\epsilon$ for sufficiently small $\epsilon > 0$, unless $P = NP$ [BP89, CT96].

With the above mentioned results on the approximation of the Steiner tree problem, the algorithm we propose in this paper improves the approximation ratio of the CND problem to 3.55 for the non-uniform demand case. We note that any improvements in the approximation of the Steiner tree problem will be reflected in the approximation ratio of our algorithm.

The problem we study is also related to the *capacitated MST problem (c-MST)* [Pap78, AG88, G91, KB83, CL73, S83]: Given an undirected edge-weighted graph with a root node and a positive integer U, the problem is to find a minimum spanning tree such that every subtree of the root node has a total demand of at most U. This problem has been cited by Kershenbaum and Boorstyn [KB83] as well as later by Gavish [G91] to model the local access network design problem when traffic is routed from a set of sources to a sink node under limited capacity of links. In c-MST, network links have a fixed capacity U, whereas in our single cable problem capacity can be purchased in multiples of U and the multiplicity of cables on each link is a decision variable. While a tree is required as a solution in c-MST, in most applications the actual requirement is to send the demand of each source node via a single path to the sink node, which is known as the non-bifurcating requirement for the demands. Our single cable problem enforces the non-bifurcating requirement without requiring that the solution be a tree. As a result, a solution to our problem may contain cycles and the total traffic going through a node may exceed U, as opposed to a solution of c-MST. We can contrast our approximation results with the best known approximation results for the c-MST. Altinkemer and Gavish [AG88] gave a 4-approximation for the non-uniform demand case and a 3-approximation for the uniform demand case of c-MST. In the non-uniform demand case, our $(\rho_{ST}+2)$-approximation has a performance ratio less than 4, and handles the Steiner version that does not require all non-sink nodes be source nodes (i.e., other Steiner nodes in the graph are allowed). When all non-sink nodes are sources performance ratio improves to 3, and to 2 for the corresponding uniform demand case.

In the next two sections, we present the algorithms for the cases of uniform and non-uniform demands and analyze their worst-case performance. We conclude with an extension of the local access design problem.

2 Uniform Demand

In this section we consider the case when every source node has the same demand value; without loss of generality we assume that the demand equals one for each source. The interesting cases for the problem arise when $U > 1$ because if $U = 1$ sending every demand to the sink through the path with minimum total cost is an optimal solution. Therefore, we analyze the problem assuming $U > 1$. We first give an outline of the proposed algorithm. We construct a Steiner tree T with terminal set $S \cup \{t\}$ and cost c_e on each edge e in polynomial time such that the cost of the tree is at most ρ_{ST} times the optimal Steiner tree cost. We also find a path from each node to the sink in G with minimum total cost. Next, we identify a subtree of T such that the total demand in the subtree equals the cable capacity U. In this subtree we select a node with the smallest cost of reaching the sink as the hub node (ties can be broken arbitrarily). We route the demand of each source to the hub node in T and send the aggregated demand at the hub to the sink via a minimum cost path in the input graph G. We repeat these steps until no more such subtrees can be identified. Finally, we cancel flow in opposite directions of an edge and reassign the source nodes to the hub nodes.

In order to describe the algorithm we need to define some more notation. The minimum cost of a path from node v to the sink t in G is denoted by $c(v,t)$. In the Steiner tree T, we designate the sink node t as the *root* of the tree and define the *level* of a node as the number of edges on its path to the sink node t. Any node adjacent to v with level one larger than the level of v is called a *child* of v. For each node v, T_v denotes the subtree of T rooted at v. At a given iteration of the algorithm, R is the set of unprocessed source nodes and $D(T_v)$ is the total unprocessed demand in T_v. That is, $D(T_v) = \sum_{s_i \in R \cap T_v} dem_i = |R \cap T_v|$ since $dem_i = 1$ for all i in the uniform case. Note that here we omit the iteration number for simplicity of notation.

The Algorithm UNIFORM, given below, takes T as input, and outputs 1) for each source, a route through which all of its demand is sent to the sink, and 2) the number of cables that are installed on the edges of the input network to support the flow of traffic going through them.

Algorithm UNIFORM:
Initialize: $R := S$
Main step:
- Pick a node v such that $D(T_v) \geq U$ and level of v is maximum.
- If no such node exists (that is, $D(T_t) < U$) or $v = t$, then go to the final step.
- Find a node, say w, in $R \cap T_v$ such that $c(w,t)$ is minimum.
- Designate w as a "hub" node and set $C = \{w\}$.
- Collect U-1 additional source nodes into C (*described below*).
- Assign these sources to w.
- Route the demand of each source in C to the hub node w via the unique paths in T.

Route the demand of C aggregated at the hub via the minimum cost path to the sink in G.
Install one copy of the cable on this path.
Remove C from R.
If R is not empty, repeat the main step.
If R is empty, go to the final step.

Final step:

If R is not empty, then
 Designate t as the hub node for sources in R
 Route all the demand in R to t via the unique paths in T.
For all edges e of T
 If the sum of flow on e in both directions exceeds U
  Cancel maximal amount of flow in opposite directions
  Reassign the sources whose flow has been cancelled (*described below*).
Install one copy of cable on the edges of T which have positive flow.

The following procedure is used to collect unprocessed source nodes from T_v in the set C at the main step of the algorithm. The set C contains only the hub node w when the procedure is called. At the end of the procedure C contains U nodes.

Procedure to collect source nodes of T_v

Add v to C, if $v \in R$.
Let $v_1, \ldots, v_k$ be the children of v.
If $w \neq v$, then
 Let v_p be the child of v such that the hub node w is in T_{v_p}.
 Add $T_{v_p} \cap R$ to C.
While $|C| < U$,
 Pick an unprocessed child of v, say v_i.
 If $D(T_{v_i}) + |C| \leq U$, then
 Add $T_{v_i} \cap R$ to C.
 Else, (T_{v_i} is collected partially)
  Scan T_{v_i} depth-first and add sources in $R \cap T_{v_i}$ to C one at a time until $|C| = U$.
Return C.

The procedure to cancel flow on an edge e of T is called after all the demand has been routed. We denote the set of sources sending flow on e in the direction towards t by S_{out}. These sources have total flow f_{out} and we will show that they send their flow to a common hub node w_{out}. Similarly, let f_{in} be the flow on e in the opposite direction with source set S_{in} assigned to the hub w_{in}.

Procedure to cancel flow on an edge e

If $f_{in} \leq f_{out}$, then cancel f_{in} as follows:
 Assign all sources of S_{in} to w_{out} and route their flow to w_{out}.
 Assign f_{in} number of sources from S_{out} to w_{in} and route their flow to w_{in}.
If $f_{in} > f_{out}$, then cancel f_{out} as follows:
 Assign all sources of S_{out} to w_{in} and route their flow to w_{in}.
 Assign f_{out} number of sources from S_{in} to w_{out} and route their flow to w_{out}.

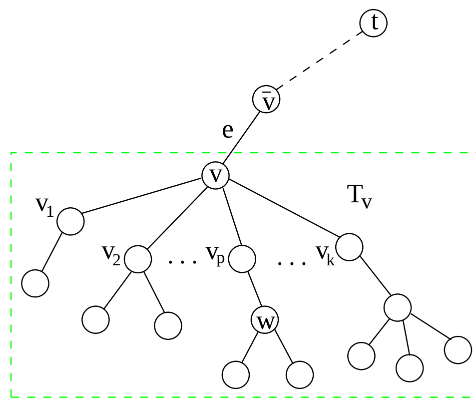

Figure 1: A node v of T, its children and its subtree.

It can be easily verified that Algorithm UNIFORM has a polynomial time complexity. We proceed to analyze the worst-case performance of the solutions output by the algorithm.

Lemma 2.1 *In a solution output by Algorithm UNIFORM, the total flow on any edge of the tree T is at most the cable capacity U.*

Proof: Let us pick an arbitrary edge e of T, and let v be the incident node on e with higher level (see Figure 1). Consider the solution output by the algorithm. In this solution, the total flow on e equals the sum of the flow coming out of T_v and the flow going into T_v. Our proof is based on these two claims:
Claim 1: The total flow going out of T_v is at most $U - 1$.
Claim 2: The total flow coming into T_v is at most $U - 1$.
To prove claims 1 and 2, we consider two cases based on how the sources in T_v are assigned to hub nodes by the algorithm. A *partially assigned subtree* has at least one of its source nodes collected in a set C and has at least one source node not in C.

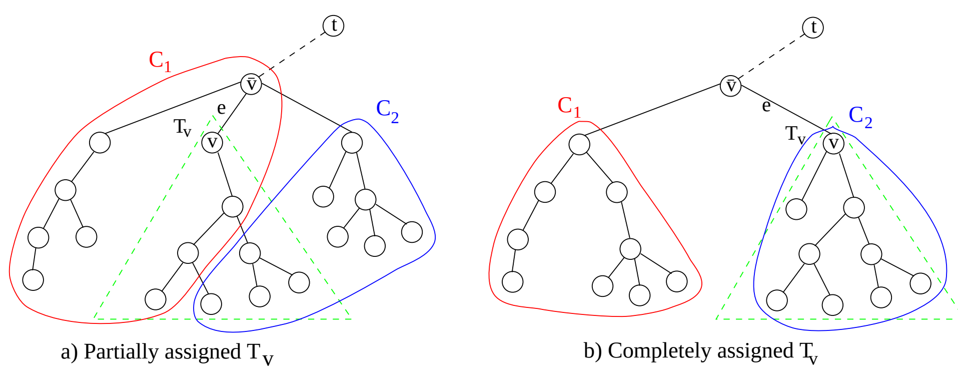

Figure 2: Examples of partially and completely assigned subtrees.

Suppose T_v is partially assigned (see Figure 2). There exists outflow from T_v, if at least one source in T_v is assigned to a hub node out of T_v. When the algorithm assigns sources in T_v to a hub node outside T_v for the first time, a subtree $T_{\bar{v}}$ with $\bar{v}$ at a smaller level than v is being processed by the algorithm. Due to the subtree selection rule, we can conclude that the total unprocessed demand in T_v is strictly less than U. Therefore, the total outflow from T_v will be at most $U - 1$. Hence, Claim 1 holds in this case.

The reason Claim 2 holds for a partially assigned T_v is as follows. When there exists an inflow into T_v, the flow is accumulated at a hub node in T_v. Since the algorithm accumulates a flow of exactly U at any hub node, a flow of at most $U - 1$ will go into T_v. The algorithm first picks a subtree and a hub node in it, and collects demand starting with the subtrees of T_v. Therefore, the algorithm will not collect sources out of T_v, unless all the sources in T_v have already been collected. This implies that once flow enters T_v, none of the nodes in T_v will become a hub node again. Thus Claims 1 and 2 are proved for the case of partially assigned T_v.

Now let us assume that T_v is not partially assigned. Then, all the sources in T_v are collected in the same set by the algorithm. If these sources are routed to a hub node out of the subtree, then the outflow is at most $U-1$. If the sources are routed to a hub node in the subtree, then the inflow is at most $U - 1$. Inflow or outflow on this edge occurs only once throughout the iterations of the algorithm. Thus, Claims 1 and 2 hold in this case, too.

For any edge of T, the flow in one direction does not exceed U, by Claims 1 and 2. If the edge e has flow in both directions such that $f_{in}(e) + f_{out}(e) > U$, then at the final step of the algorithm we cancel flow of equal value in opposite directions. As a result, the flow in one direction will be equal to zero, and the flow in the opposite direction will be equal to the difference of $f_{in}(e)$ and $f_{out}(e)$; hence, the total flow on e will not exceed U. The cancellation of flow of value $k = |f_{in}(e) - f_{out}(e)|$ will lead to the switching of the hub nodes of k sources which sent flow on e in one direction with that of k other sources which sent flow in the opposite direction. As a result, the routes for these sources will change but the total flow on any of the edges on their routes will not increase. Consider the subtree composed of the union of routes of S_{in} and S_{out}, which we call T_{in} and T_{out}. For the edges in the intersection of T_{in} and T_{out}, flow in opposite directions will be cancelled; hence the total flow on these edges will decrease. For the remaining edges of T_{in} and T_{out}, some of the flow in one direction (of value up to k) will reverse its direction but the total flow will not increase in amount. An example to flow cancellation is given in Figure 3. □

Theorem 2.2 *Algorithm UNIFORM is a $(1 + \rho_{ST})$-approximation algorithm for the capacitated network design problem with uniform demand.*

Proof: Let C_{OPT} be the cost of an optimal solution and C_{HEUR} be the cost of a solution output by Algorithm UNIFORM. Let C_{ST} denote the cost of the cables installed on the edges of the Steiner tree T (whose cost is at most ρ_{ST} times the cost of an optimal Steiner tree). Let C_G be the cost of cables installed on the edges of G to send aggregated flow from the hub nodes to the sink node with minimum cost.

By Lemma 2.1, at most one copy of cable is sufficient to accomodate flow on the edges of the Steiner tree T. The cost of a Steiner tree with terminal set $S \cup \{t\}$ is a lower bound on the optimal cost because we must connect the nodes in S to t and install at least one copy of the cable on each connecting edge. Therefore, $C_{ST} \leq \rho_{ST} C_{OPT}$.

A source set C_k collected at iteration k has demand equal to U and the algorithm installs one copy of the cable on the path from the hub node w_k to the sink t with cost $c(w_k, t)$. The term

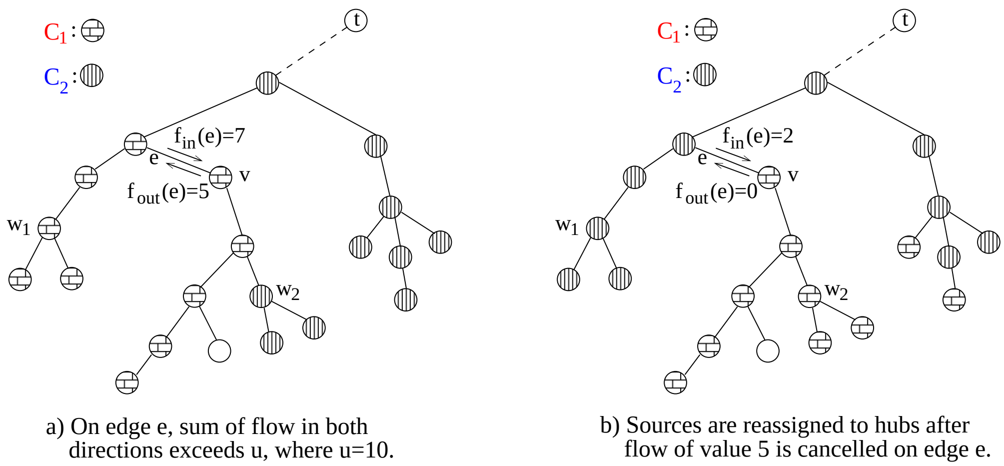

Figure 3: An example of cancelling flow and reassigning sources to hub nodes. Here w_1 and w_2 are hub nodes chosen in the order of their indices, with collected source sets C_1 and C_2, respectively.

$\sum_{s_i \in C_k} \frac{c(s_i,t)}{U}$ is the cost associated with routing the unit demand of each source s_i of C_k through the minimum cost path and purchasing on each edge of the path $1/U$ fraction of the cable. The algorithm identifies a collection of subtrees with disjoint source sets (but possibly common non-source nodes). Therefore, $\sum_k \sum_{s_i \in C_k} \frac{c(s_i,t)}{U}$ is a lower bound on C_{OPT}. Since the algorithm sends the total demand of a set C_k via the source in C_k with the minimum cost of reaching t (that is, the hub node w_k) and $|C_k| = U$, we get

$$c(w_k, t) = \min_{s_i \in C_k} c(s_i, t) \leq \frac{\sum_{s_i \in C_k} c(s_i, t)}{|C_k|} = \sum_{s_i \in C_k} \frac{c(s_i, t)}{U}.$$

Thus, we finally have

$$C_G = \sum_k c(w_k, t) \leq \sum_k \sum_{s_i \in C_k} \frac{c(s_i, t)}{U} \leq C_{OPT}.$$

Therefore, $C_{HEUR} = C_{ST} + C_G \leq (1 + \rho_{ST}) C_{OPT}$. □

When every non-sink node in the input graph has a non-zero demand, we do not need to construct a Steiner tree. Instead, we input a minimum cost spanning tree T to algorithm UNIFORM. In the proof of 2.2, C_{ST} will refer to the cost of the spanning tree T and ρ_{ST} will be equal to 1. Therefore, we obtain a performance ratio of 2.

Corollary 2.3 *Algorithm UNIFORM is a 2-approximation algorithm for the capacitated network design problem with uniform demand at every non-sink node.*

3 Non-uniform Demand

When the demand of a source node can be any positive integral value, it is no longer possible to collect sources with total demand exactly equal to the capacity U. Suppose we are allowed to split the demand of a source into any integral units each of which can be routed in separate paths to the sink. In that case, the algorithm of the previous section can be used by expanding each source s_i with demand dem_i to a set of sources with unit demand connected by zero-length edges in the tree. However, in the more general case, all the flow of a source must use the same path to the sink. In this case, we modify Algorithm UNIFORM so that we send demand directly to the sink when it accumulates to an amount between $U/2$ and U. To guarantee that we don't exceed U while collecting demand, we send all sources with demand at least $U/2$ directly at the beginning of the algorithm.

For a source set C, let $dem(C)$ be the total demand of sources in C. As defined for the uniform demand case, we use $D(C)$ to denote the total *remaining* (unprocessed) demand of C. The modified algorithm, which we call Algorithm NON-UNIFORM, is given below.

<u>Algorithm NON-UNIFORM:</u>
Initialize: $R := S$.
Preprocessing: (send large demands directly)
  For all sources s_i such that $dem_i \geq U/2$,
    Route the demand to the sink via the minimum cost path in G.
    Install $\lceil \frac{dem_i}{U} \rceil$ copies of cable on the edges of this $s_i - t$ path.
    Remove s_i from R.
Main step:
  Pick a node v such that $D(T_v) \geq U/2$ and level of v is maximum.
  If no such node exists (that is, $D(T_t) < U/2$) or $v = t$, then go to the final step.
  Find a node, say w, in $R \cap T_v$ such that $c(w,t)$ is minimum.
  Designate w as a "hub" node and set $C = \{w\}$.
  Collect additional source nodes in C (*described below*).
  Assign these sources to w.
  Route demand of each source in C to the hub node w via the unique path in T.
  Route demand of C aggregated at w via the minimum cost $w - t$ path in G.
  Install one copy of cable on the edges of this $w - t$ path.
  Remove C from R and set $C = \emptyset$.
  If R is not empty, repeat the main step.
  If R is empty, go to the final step.
Final step:
  If R is not empty, then route all the demand in R to t via the unique paths in T.
  Install one copy of cable on the edges of T which have positive flow.

The following procedure is used to collect unprocessed source nodes from T_v in the set C at the main step of the algorithm. The set C contains only the hub node w when the procedure is called. At the end of the procedure demand of C is in the range $[U/2, U]$.

<u>Procedure to collect source nodes of T_v:</u>
Add v to C, if $v \in R$.
Let $v_1, \ldots, v_k$ be the children of v.
If $w \neq v$, then
    Let v_p be the child of v such that the hub node w is in T_{v_p}.

Add $T_{v_p} \cap R$ to C.
While $dem(C) < U/2$,
Pick an unprocessed child of v, say v_i.
Add $T_{v_i} \cap R$ to C.
Return C.

Lemma 3.1 *Algorithm NON-UNIFORM outputs a solution in which 1) flow on any edge of the tree T is at most U, and 2) flow sent from a hub node to the sink in G at each iteration of the algorithm is at least $U/2$ and at most U.*

Proof: The proof is simpler compared to the uniform-demand case because the algorithm does not assign any subtree partially. Consider an edge e of T. Let v be incident on e such that e is not in T_v. Since all the sources in T_v are collected in the same set by the algorithm, the demand of these sources is routed to a hub node either out of the subtree, or in the subtree, but not both. Thus, flow on e exists only in one direction. If the demand of sources is routed to a hub node out of T_v, then outflow is at most $U-1$. If the demand is routed to a hub node in the subtree, then inflow is at most $U-1$. Thus, for any edge of T, flow does not exceed U.

Due to the subtree selection rule in the algorithm, if a subtree T_v is selected at an iteration, then all the subtrees rooted at its children have remaining demand strictly less than $U/2$. Therefore, when source nodes are collected in C, the first time the total collected demand $dem(C)$ exceeds $U/2$, $dem(C)$ will be at most U. Therefore, the total flow sent from the hub to the sink at this iteration is in the range $[U/2, U]$. □

Theorem 3.2 *Algorithm NON-UNIFORM is a $(2+\rho_{ST})$-approximation algorithm for the capacitated network design problem with non-uniform demand.*

Proof: We use the same definitions of C_{OPT}, C_{HEUR}, C_G and C_{ST} as in the proof of Theorem 2.2.

By Lemma 3.1, at most one copy of the cable is sufficient to accommodate flow on the edges of the Steiner tree T. Therefore, $C_{ST} \leq \rho_{ST} C_{OPT}$.

For a source set C_k collected at iteration k, the algorithm installs one copy of the cable on the minimum cost $w_k - t$ path in G. By Lemma 3.1, at most one copy of cable is sufficient to accommodate flow on this path and the flow utilizes at least half of the capacity of the cable. Thus, $\sum_{s_i \in C_k} dem_i \geq U/2$. If we were allowed to pay for only the portion of the cable used, then a minimum cost solution would use the minimum cost path from each source to the sink. This solution would have cost $\sum_{s_i \in S} \frac{dem_i}{U} \cdot c(s_i, t)$ and this is a lower bound on C_{OPT}. Since source sets collected by the algorithm have disjoint sources, and the demand from a set C_k is sent via the source in C_k that is closest to t (the hub node w_k),

$$C_{OPT} \geq \sum_k \sum_{s_i \in C_k} \frac{dem_i}{U} c(s_i, t) \geq \sum_k \sum_{s_i \in C_k} \frac{dem_i}{U} (\min_{s_i \in C_k} c(s_i, t)) \geq \sum_k \sum_{s_i \in C_k} \frac{1}{2} c(w_k, t) = \frac{1}{2} C_G.$$

The last inequality follows since $\sum_{s_i \in C_k} dem_i \geq \frac{U}{2}$ and $\min_{s_i \in C_k} c(s_i, t) = c(w_k, t)$. Therefore, $C_{HEUR} = C_{ST} + C_G \leq (2 + \rho_{ST}) C_{OPT}$. □

When every non-sink node in the input graph has a non-zero demand, we do not need to construct a Steiner tree. Instead, we input a minimum cost spanning tree T to algorithm NON-UNIFORM. In the proof of Theorem 3.2, C_{ST} will refer to the cost of the spanning tree T and ρ_{ST} will be equal to 1. Therefore, we obtain a performance ratio of 3.

Corollary 3.3 *Algorithm NON-UNIFORM is a 3-approximation algorithm for the capacitated network design problem with non-uniform demand at every non-sink node.*

4 Extensions

Our methods apply to the following extension of the local access network design problem. Instead of specifying a single sink node, any node v in the graph can be used as a node that sinks U units of demand at a cost of f_v. A node is allowed to sink more than U units of demand by paying $\lceil \frac{dem}{U} \rceil \cdot f_v$ cost to sink dem units of flow. The problem is to open sufficient number of sinks and route all the demands to these sinks at minimum cable plus sink opening costs.

To model this extension, we extend the metric in two steps: 1) create a new sink node t with edges to every vertex v of cost f_v, 2) take the metric completion of this augmented network. Notice that the second step may decrease some of the costs on the edges incident on the new sink t (e.g., if $f_i + c(j,i) < f_j$, then the cost of the edge (j,t) can be reduced from f_j to $f_i + c(j,i)$), or between any pair of original nodes (e.g., if $c(i,j) > f_i + f_j$, then we may replace the former by the latter). Bearing this in mind, it is not hard to see that any solution in the new graph to the single cable problem with t as the sink and with the modified costs can be converted to a solution to the original problem of the same cost. Thus, our algorithms in the previous sections apply to give the same performance guarantees.

References

[AG88] K. Altinkemer and B. Gavish, "Heuristics with constant error guarantees for the design of tree networks," Management Science, **34**, 331-341, 1988.

[AZ98] M. Andrews and L. Zhang, "The access network design problem," In Proc. of the 39th Ann. IEEE Symp. on Foundations of Computer Science, 42-49, October 1998.

[BP89] M. Bern and P. Plassmann, "The Steiner tree problem with edge lengths 1 and 2," Information Processing Letters, **32**, 171-176, 1989.

[BR94] P. Berman and V. Ramaiyer, "Improved approximations for the Steiner tree problem," J. of Algorithms, **17**, 381-408, 1994.

[CL73] K. M. Chandy and T. Lo, "The capacitated minimum tree," Networks, **3**, 173-182, 1973.

[CT96] A. Clementi and L. Trevisan, "Improved Non-approximability results for minimum vertex cover with density constraints," Proceedings of 2nd Computing and Combinatorics Conference. COCOON'96, Berlin, Springer Verlag, 333-342,1996.

[DMM97] M. Dell'Amico, F. Maffioli and S. Martello, Editors, "Annotated Bibliographies in Combinatorial Optimization," Wiley-Interscience Publication, 1997.

[G91] B. Gavish, “Topological design of telecommunication networks - local access design methods,” Annals of Operations Research, **33**, 17-71, 1991.

[GJ77] M. R. Garey and D. S. Johnson, “The rectilinear Steiner tree problem is NP-complete,” SIAM J. of Applied Math., **32**, 826-834, 1977.

[GKK+01] N. Garg, R. Khandekar, G. Konjevod, R. Ravi, F. S. Salman, and A. Sinha, “On the Integrality Gap of a Natural Formulation of the Single-Sink Buy-at-Bulk Network Design Problem,” In Proc. of the 8th International Integer Programming and Combinatorial Optimization Conference, LNCS 2081,170-184, 2001.

[HP99] S. Hougardy and H.J. Prömmel, “A 1.598 approximation algorithm for the Steiner problem in graphs”, Proc. of the 9th Ann. ACM-SIAM Symp. on Discrete Algorithms, 448-453, 1999.

[KB98] Kawatra, R. and D. L. Bricker, ”A multiperiod planning model for the capacitated minimal spanning tree problem”, European Journal of Operational Research, **121**, 412-419, 2000.

[KB83] A. Kershenbaum and R. Boorstyn, “Centralized teleprocessing network design,” Networks, **13**, 279-293, 1983.

[KRY93] S. Khuller, B. Raghavachari and N. E. Young, “Balancing minimum spanning and shortest path trees,” Algorithmica, **14**, 305-322, 1993.

[KZ97] M. Karpinsky and A. Zelikovsky, “New approximation algorithms for the Steiner tree problem,” J. Combinatorial Optimization, **1**, 47-65, 1997.

[MP94] Y. Mansour and D. Peleg, “An approximation algorithm for minimum-cost network design,” The Weizman Institute of Science, Rehovot, 76100 Israel, Tech. Report CS94-22, 1994; Also presented at the DIMACS workshop on Robust Communication Networks, 1998.

[Pap78] C. H. Papadimitriou, “The complexity of the capacitated tree problem,” Networks, **8**, 217-230, 1978.

[PS97] H.J. Prömmel and A. Steger, “RNC-approximation algorithms for the Steiner problem”, Proc. of the 14th Ann. Symp. on Theoretical Aspects of Computer Science, 559-570, 1997.

[RZ00] G. Robins and A. Zelikovsky, “Improved steiner tree approximation in graphs”, Proc. of the 10th Ann. ACM-SIAM Symp. on Discrete Algorithms, 770-779, 2000.

[SCR+97] F.S. Salman, J. Cheriyan, R. Ravi and S. Subramanian, “Approximating the single-sink link-installation problem in network design,” SIAM J. Optimization, 595-610, 2000.

[S83] R. L. Sharma, “Design of an economical multidrop network topology with capacity constraints,” IEEE Trans. Comm., **31**, 590-591, 1983.

[SS99] B. Sanso and P. Soriano, Editors, “Telecommunications Network Planning,” Kluwer Academic Publishers, 1999.

[TM80] H. Takahashi and A. Matsuyama, “An approximate solution for the Steiner problem in graphs”, Math. Japan, **24**, 573-577, 1980.

[Z93] A. Zelikovsky, “An 11/6-approximation algorithm for the network Steiner problem,” Algorithmica, **9**, 463-470, 1993.