Improved Approximation Algorithms for the Single-Sink Buy-At-Bulk Network Design Problems

Raja Jothi and Balaji Raghavachari

University of Texas at Dallas, Richardson, TX 75083, USA
{raja, rbk}@utdallas.edu

Abstract. Consider a given undirected graph $G = (V, E)$ with non-negative edge costs, a root node $r \in V$, and a set $D \subseteq V$ of *demands* with d_v representing the units of flow that demand $v \in D$ wishes to send to the root. We are also given K types of cables, each with a specified capacity and cost per unit length. The *single-sink buy-at-bulk* (SSBB) problem asks for a low-cost installation of cables along the edges of G, such that the demands can simultaneously send their flows to sink/root r. The problem is studied with and without the restriction that the flow from a node must follow a single path to the sink (indivisibility constraint). We are allowed to install zero or more copies of a cable type on each edge. The SSBB problem is NP-hard. In this paper, we present a 145.6-approximation for the SSBB problem improving the previous best ratio of 216. For the *divisible* SSBB (DSSBB) problem, we improve the previous best ratio of 72.8 to α_K, where α_K is less than 65.49 for all K. In particular, $\alpha_2 < 12.7, \alpha_3 < 18.2, \alpha_4 < 23.8, \alpha_5 < 29.3$, $\alpha_6 < 33.9$.

1 Introduction

Consider a given undirected graph $G = (V, E)$ with non-negative edge costs, a root node $r \in V$, and a set $D \subseteq V$ of *demands* with d_v representing the units of flow that demand $v \in D$ wishes to send to the root. We are also given K types of cables, each with a specified capacity and a cost per unit length. The cost per unit capacity per unit length of a high-capacity cable is typically less than that of a low-capacity cable, reflecting "economy of scale". In other words, it is cheaper to buy a cable of larger capacity than many cables (adding up to same capacity) of smaller capacity. The extensively studied *single-sink buy-at-bulk* (SSBB) problem, also known as the *single sink edge Installation* problem, asks for a low-cost installation of cables along the edges of G, such that the demands can simultaneously send their flows to sink/root r, under the restriction that the flow from a node must follow a single path to the sink (indivisibility constraint). We are allowed to install zero or more copies of a cable type on each edge. By *divisible* SSBB (DSSBB) problem, we refer to the version of the SSBB problem without the indivisibility constraint.

The SSBB problem has applications in the hierarchical design of telecommunication networks, in which the traffic from a source must follow a single path

to the sink. The DSSBB problem has its own applications: a classic application would be that of routing oil from several oil wells to a major refinery [8].

The *buy-at-bulk* network design problem was introduced by Salman, Cheriyan, Ravi and Subramanian [8]. They showed that the problem is NP-hard, by showing a simple reduction from the Steiner tree problem or the knapsack problem. The problem remains NP-hard even when only one cable type is available. They also presented a $O(\log n)$-approximation algorithm for the SSBB problem in Euclidean graphs. For problem instances in general metric spaces, Awerbuch and Azar [1] presented a $O(\log^2 n)$-approximation algorithm. Their algorithm works even for multi-root version of the problem. Bartal's tree embeddings [2] can be used to improve their ratio to $O(\log n \log\log n)$. Garg et al. [3] presented an $O(K)$-approximation algorithm based on LP-rounding. Guha, Meyerson and Munagala [4] presented the first constant-factor approximation algorithm, whose ratio was estimated to be around 2000 by Talwar [9]. In the same paper, Talwar presented an LP-based rounding algorithm with an improved ratio of 216.

Recently, Gupta, Kumar and Roughgarden [5] presented a simple and elegant 72.8-approximation algorithm for the SSBB problem. But unfortunately, their approach does not guarantee that the flow from a node follow a single path to the sink. In other words, their ratio of 72.8 holds for the DSSBB problem, but not for the SSBB problem. That leaves Talwar's ratio of 216 as the current best for the SSBB problem.

In this paper, we design a 145.6-approximation algorithm for the SSBB problem, using ideas from Gupta, Kumar and Roughgarden [5], but guaranteeing the indivisibility constraint is not straightforward. We introduce a new "redistribution" procedure which is pivotal in guaranteeing that the flow from a source follows a single path to the sink. We also propose a modification to their DSSBB algorithm that reduces the ratio from 72.8 to α_K, where α_K is less than 65.49 for all K. In particular, $\alpha_2 < 12.7, \alpha_3 < 18.2, \alpha_4 < 23.8, \alpha_5 < 29.3, \alpha_6 < 33.9$.

2 Preliminaries

Let $G = (V, E)$ be the input graph with $D \subseteq V$ being the set of demands. We use the terms vertices and nodes interchangeably. Also, depending up on the context, we use the term "demand" to denote a vertex or the flow out of it. Let c_e denote the length of edge e. We also use c_{xy} to denote the length of an edge connecting nodes x and y. We use the metric completion of the given graph. Let u_i and σ_i denote the capacity and cost per unit length of cable type i. We define $\delta_i = \sigma_i/u_i$ to be the "incremental cost" of using cable type i. The value of δ_i can also interpreted as the cost per unit capacity per unit length of cable type i. Let us assume that each u_i and σ_i (and by definition δ_i) is a power of $1+\epsilon$, $\epsilon > 0$, which can be enforced by rounding each capacity u_i down to the nearest power of $1+\epsilon$, and each σ_i up to the nearest power of $1+\epsilon$. This assumption is not without loss of generality, and can be accounted by losing a factor of $(1+\epsilon)^2$ in the approximation ratio. We will choose ϵ later. This idea of rounding is derived from [5], where they used powers of 2, thus effectively choosing ϵ to be 1.

The following properties on the costs and capacities of cable types have been known [4, 5]. Without loss of generality, assume that the cables are ordered such that $u_i < u_j$ and $\sigma_i < \sigma_j$ for all $i < j$. Note that if $u_i \leq u_j$ and $\sigma_i \geq \sigma_j$, then we can eliminate cable type i from consideration. We can also assume that $u_1 = \sigma_1 = 1$, as this can be obtained by appropriate scaling, though it may leave non-integer weights at vertices. For each $j < k$,

$$\frac{\sigma_k}{u_k} < \frac{\sigma_j}{u_j}. \tag{1}$$

Otherwise, cable type k can be replaced by u_k/u_j copies of cable type j without increase in cost. The fact that $\delta_j = \sigma_j/u_j$ is a power of $1+\epsilon$ implies that $\delta_{j+1} \leq \delta_j/(1+\epsilon)$ for all j, since $u_{j+1} \geq (1+\epsilon)u_j$. Let $g_k = \frac{\sigma_{k+1}}{\sigma_k} u_k$. By equation (1),

$$1 = u_1 < g_1 < u_2 < g_2 < \ldots < u_k < g_k = \infty.$$

Since σ_i is a power of $1+\epsilon$ for any i, and $\sigma_{j+1} > \sigma_j$, using equation (1) we get,

$$\frac{u_{j+1}}{u_j} > 1 + \epsilon.$$

Let OPT denote an optimal solution with cost $C^* = \sum_j C^*(j)$, where $C^*(j)$ is the amount paid for cable type j in OPT. We state the following lemma and its proof from [5], as its understanding is crucial for an easier understanding of our algorithms.

Lemma 1 (Redistribution Lemma [5]). *Let T be a tree rooted at r with each edge having capacity U. For each vertex $j \in T$, let $w(j) < U$ be the weight located at j with $\sum_j w(j)$ a multiple of U. Then there is an efficiently computable (random) flow on the tree that redistributes weights without violating edge capacities, so that each vertex receives a new weight $w'(j)$ that is either 0 or U. Moreover,*

$$\mathbf{Pr}\ [\ w'(j) = U\ ] = w(j)/U$$

Proof. Replace each edge in T with two oppositely directed arcs. Let Y be a value chosen uniformly at random from $(0, U]$. Take an Euler tour of the vertices in T, starting from r and visiting all the other vertices $\{j_1, j_2, \ldots, j_m\}$ in T. Let a counter Q be set to 0 initially. On visiting vertex j_k, we update $Q \leftarrow Q + w(j_k)$. Also, let Q_{old} and Q_{new} be the value of v just before and after visiting j_k, respectively. On visiting j_k, if $xU + Y \in (Q_{old}, Q_{new}]$ for some integer x, then "mark" j_k and ask that it send $Q_{new} - (xU + Y)$ weight to the next marked vertex lying clockwise on the tour. In the other case, we ask that j_k send all its weight to the next marked vertex lying clockwise on the tour. This construction ensures that the maximum flow on an directed edge is at most U, and that the probability that a vertex j gets marked is $w(j)/U$, which is exactly the probability that j receives a weight of U.

Since we were working on a directed tour, the cost of this redistribution is at most twice the cost of the tree T, as an edge in T was replaced by two oppositely directed arcs. But, using simple flow canceling argument, one can show that one copy of the edges in T is sufficient for such a redistribution. ∎

3 The Algorithms

We first show how to modify the algorithm of Gupta, Kumar and Roughgarden [5] to obtain an approximation ratio of α_K, where α_K is less than 65.49 for all K. Recall that their algorithm solves the DSSBB problem, and not the SSBB problem. We then present our main result, an approximation algorithm for the SSBB problem that achieves an approximation ratio of 145.6 in Section 3.2.

3.1 The DSSBB Problem

The vertices of the graph $G = (V, E)$ may have non-integer weights as demands, because of the scaling done to make $u_1 = \sigma_1 = 1$. Since the flow is divisible, there is no loss of generality in assuming that $d_j \leq 1$, because a demand greater than 1 can be split into multiple demands by splitting a vertex into $\lceil d_j \rceil$ vertices. The algorithm is simpler with this assumption, and easily adapts to higher demands by adjusting the probabilities without actually splitting vertices.

Construct a ρ-approximate Steiner tree T_0, using cables with capacity $u_1 = 1$, spanning all the demands in D. Redistribute the demands using the construction in the proof of Lemma 1, with $U = 1$, and collect *integral* demands at some subset of vertices in D. The cost incurred to do this redistribution is just the cost of the Steiner tree [5], and since the optimal solution contains a candidate Steiner tree, we incur a cost of at most $\rho \times \sum_j C^*(j)/\sigma_j$. We can assume that the number of demands $|D|$ is a power of $1 + \epsilon$, as this can be achieved by placing *dummy* demands at the root vertex r.

The algorithm given below closely follows the incremental design of Gupta et al.'s algorithm [5] to build the network. The algorithm proceeds in stages using only cable types t and $t + 1$ in stage t, except for the last stage ($t = K$) in which only cables of type K are used.

At the beginning of the first stage, $D_1 = D$ with each demand $j \in D$ having weight $d_j = 1 = u_1$. In general, at the beginning of stage t, D_t is a set of $|D|/u_t$ vertices, each with demand u_t. During stage t, our algorithm (presented below) uses u_{t+1} as the "aggregation threshold" to combine several demands of weight u_t into a single demand of weight u_{t+1}. Unlike [5], where capacities are powers of 2 which ensures that u_{t+1} is an integral multiple of of u_t, in our algorithm u_{t+1} is not necessarily an integral multiple of u_t. As a result, during the aggregation process our algorithm may have to combine demands of weight u_t from, say, 1.33 vertices to obtain a demand of weight u_{t+1}. The cables required to perform such an aggregation are bought by the algorithm. The demand will reach the root at the end of the algorithm. The final solution is then given by the union of all the paths used in the aggregation stages.

Given below are the steps performed at stage t of the algorithm. Its first three steps are exactly the same as in [5]. Whenever we mention a fraction of a vertex, we mean a fraction of the demand from that vertex.

D1. *Mark* each demand in D_t with probability $p_t = u_t/g_t$, and let D_t^* be the marked demands.

D2. Construct a ρ-approximate Steiner tree T_t on $F_t = D_t^* \cup \{r\}$. Install a cable of type $t+1$ on each edge of this tree.

D3. For each vertex $j \in D_t$, send its weight u_t to the nearest member of F_t using cables of type t. Let $w_t(i)$ be the weight collected at $i \in F_t$. All vertices that sent their demands to the same vertex i are said to belong to i's *family*, which we call as G_i.

D4. A vertex $i \in F_t$ collects the demands sent to it by all vertices in its family, G_i, divides it into groups of size u_{t+1}. Each member of G_i may partition its flow and contribute to at most two groups. Flow from a group g is sent back to a random vertex of g by buying a new cable of type $t+1$. If the whole vertex belongs to g, then the probability that that vertex receives back a weight of u_{t+1} is u_t/u_{t+1}. But if only a fractional part f of a vertex demand belongs to g, then the probability that that vertex receives back a weight of u_{t+1} is fu_t/u_{t+1}. Some *residual* demand may be left over at i which will be aggregated into demands of u_{t+1} using redistribution in the next step. Let the number of residual vertices at i be b_i.

D5. After rerouting the collected weight back from i to vertices in D_t in the above step for all $i \in F_t$, we aggregate the weights from residual vertices into groups of weight exactly u_{t+1} using Lemma 1 with $T = T_t, w_t(i) = b_i u_t$ and $U = u_{t+1}$. For every $i \in F_t$ that receives u_{t+1} weight as a result of this aggregation process, send the weight back from i to one of i's b_i residual vertices, chosen uniformly at random, using newly bought cable of type $t+1$. If the whole vertex is a residual vertex, then the probability that that residual vertex receives back the weight of u_{t+1} is $1/b_i$. But if only a fractional part f of a vertex demand is residual, then the probability that that vertex receives the weight of u_{t+1} is f/b_i. In this scheme, a vertex j may receive back a weight of $2u_{t+1}$ in stage t as a result of it being in two groups, which can be viewed as duplicating the vertex.

At stage t, since the u_{t+1} demands from $i \in F_t$ for all i are returned back to a subset of vertices in D_t, $D_{t+1} \subseteq D_t$ for all t. When $t = K$, we set $p_K = 0$. Hence, in the Kth stage of the algorithm none of the demands are marked, and thus the weights of all vertices in D_K are sent directly to root r using cables of type K. We use the following lemmas to analyze our algorithm.

The lemma below appears as Lemma 4.2 in [5]. But its proof in [5] is not directly applicable to our algorithm, because the value of u_{t+1} in our algorithm is not necessarily an integral multiple of the value of u_t, for all t.

Lemma 2. *For every non-root vertex $j \in D$ and stage t*

$$\mathbf{Pr}[j \in D_t] = 1/u_t.$$

Proof. We prove the lemma by induction on t. The lemma is clearly true for the base case, $t = 1$, since $u_t = 1$. Suppose the lemma is true for stage t. We will show that it is true for stage $t+1$. In stage t, let $j \in D_t$ send its weight to $i \in F_t$. Vertex j must satisfy one of the following conditions: (i) j belongs to just one group, (ii) j belongs to two different groups, (iii) j is fully a residual vertex, and

(iv) part of j belongs to a group while the rest of j is residual. Recall that a vertex can belong to at most 2 groups.

In case (i), the probability that j receives back the group weight of u_{t+1} is u_t/u_{t+1}. In case (ii), let a fraction f of j belong to group g_1 and the rest of j belong to group g_2. The probability that j receives back g_1's weight of u_{t+1} is fu_t/u_{t+1}, while the probability that j receives back g_2's weight of u_{t+1} is $(1-f)u_t/u_t t+1$. Overall, the probability that j receives back a weight of u_{t+1} is u_t/u_{t+1}. In case (iii), the probability that i is assigned the weight of u_{t+1} is $b_i u_t/u_{t+1}$, and the probability that j receives this weight from i is $1/b_i$, thus making the overall probability that j receives a weight of u_{t+1} to be u_t/u_{t+1}. By a similar argument, it is clear that the probability that j receives a weight of u_{t+1} in case (iv) is u_t/u_{t+1}. Thus, we conclude that

$$\begin{aligned}\mathbf{Pr}\ [\ j \in D_{t+1}\ ] &= \mathbf{Pr}\ [\ j \in D_{t+1} \mid j \in D_t\ ] \cdot \mathbf{Pr}\ [\ j \in D_t\ ] \\ &= (u_t/u_{t+1})(1/u_t) \\ &= 1/u_{t+1}.\end{aligned}$$

∎

The following lemma, proved by Gupta, Kumar and Roughgarden [5], applies to our algorithm as well. The proof involves taking all cables of higher capacity used by an optimal solution, and then extending it using randomization to span F_t, and showing that this solution has low expected cost.

Lemma 3 ([5]). *Let T_t^* be the optimal Steiner tree on F_t, and $c(T_t^*) = \sum_{e \in T_t^*} c_e$. Then*

$$\mathbf{E}\ [\ c(T_t^*)\ ] \leq \sum_{s>t} \frac{1}{\sigma_s} C^*(s) + \sum_{s \leq t} \frac{1}{\delta_s . g_t} C^*(s). \tag{2}$$

Lemma 4. *The expected cost incurred in stage t is at most $(2 + \rho + \frac{2}{1+\epsilon})$ times $\sigma_{t+1}\mathbf{E}[c(T_t^*)]$, where T_t^* is the optimal Steiner tree on F_t.*

The proof of the above lemma is given as Lemma 4.4 in [5] with $\epsilon = 1$. Cost incurred during stage t is accounted for as follows: (i) the cost of the cables to construct the Steiner tree in Step D2 is at most $\rho\sigma_{t+1}c(T_t^*)$, (ii) the cost of the cables used in Step D3 is at most $2\sigma_{t+1}c(T_t^*)$, and (iii) the cost incurred in Steps D4 and D5 to reroute the demands back to random vertices in D_t is at most

$$\left(\frac{\delta_{t+1}}{\delta_t}\right).2\sigma_{t+1}c(T_t^*) \leq \left(\frac{1}{1+\epsilon}\right).2\sigma_{t+1}c(T_t^*).$$

Theorem 1. *The approximation ratio α_K of our* DSSBB *algorithm is at most 65.49.*

Proof. Recall that by rounding the costs and capacities of cables to powers of $(1+\epsilon)$, we lost a factor of $(1+\epsilon)^2$ in the approximation ratio. We incurred a cost of

$$C_{T_0} \leq \rho \sum_j C^*(j)/\sigma_j$$

for the construction of Steiner tree T_0 to ensure integral demands at vertices. The total cost C_s incurred during the K stages of the algorithm can be obtained by substituting equation (2) in Lemma 4 and summing over all t, as shown below.

$$C_s \leq \left(2+\rho+\frac{2}{1+\epsilon}\right)\cdot\left(\sum_{t\geq 1}\frac{\delta_t}{\delta_1}C^*(1)+\sum_{s=2}^{K}\Big(\sum_{t=1}^{s-1}\frac{\sigma_{t+1}}{\sigma_s}+\sum_{t\geq s}\frac{\delta_t}{\delta_s}\Big)C^*(s)\right).$$

The cost of the final solution is given by

$$\begin{aligned} C &= \Delta(C_{T_0}+C_s) \\ &< \Delta\Big(2+\rho+\frac{2}{1+\epsilon}\Big)\left(\Big(\frac{1}{\sigma_1}+\sum_{t\geq 1}\frac{\delta_t}{\delta_1}\Big)C^*(1)+\sum_{s=2}^{K}\Big(\sum_{t=0}^{s-1}\frac{\sigma_{t+1}}{\sigma_s}+\sum_{t\geq s}\frac{\delta_t}{\delta_s}\Big)C^*(s)\right), \end{aligned}$$

where $\Delta=(1+\epsilon)^2$. Since σ_t and δ_t are powers of $1+\epsilon$, the summations are upper bounded by $1+1/\epsilon$. This simplifies the above equation to

$$C < (1+\epsilon)^2\Big(2+\rho+\frac{2}{1+\epsilon}\Big)\times 2\Big(1+\frac{1}{\epsilon}\Big)\sum_{s}C^*(s), \tag{3}$$

which when optimized for ϵ gives a ratio of 65.4899 for $\epsilon\approx 0.585735$. Here we are using the current best approximation algorithm for finding a Steiner tree, which guarantees an approximation ratio of $\rho=1+\ln(3)/2$ [7]. ∎

Corollary 1. *For a fixed K, $\alpha_2<12.7, \alpha_3<18.2, \alpha_4<23.8, \alpha_5<29.3$, $\alpha_6<33.9$ and so on.*

Proof. The cost of the final solution $C=(1+\epsilon)^2(C_{T_0}+C_s)$ can be rewritten as

$$\begin{aligned} C \leq (1+\epsilon)^2\Bigg[&\rho\sum_{s=1}^{K}\frac{1}{\sigma_s}C^*(s) \\ &+\left(2+\rho+\frac{2}{1+\epsilon}\right)\cdot\left(\sum_{t\geq 1}\frac{\delta_t}{\delta_1}C^*(1)+\sum_{s=2}^{K}\Big(\sum_{t=1}^{s-1}\frac{\sigma_{t+1}}{\sigma_s}+\sum_{t\geq s}\frac{\delta_t}{\delta_s}\Big)C^*(s)\right)\Bigg]. \end{aligned}$$

For a fixed K, there exists an $\epsilon>0$ for which the corollary can be mathematically verified. ∎

3.2 The SSBB Problem

During the preprocessing step, Gupta et al.'s algorithm [5] and our DSSBB algorithm in Section 3.1 use redistribution on T_0 to guarantee integral demands at the vertices. Later, vertices of integral demands are duplicated so that the demands at vertices are unit weight. Because of this redistribution and duplication, there is no guarantee that the demand from a vertex in the input graph travels along a single path to the sink, as the demand at a vertex may have been split

during the redistribution and/or duplication process. In our algorithm below, we make sure that demand at a vertex follows a single path to the sink. Like [5], we set $\epsilon = 1$, which makes u_i and σ_i (and by definition δ_i) powers of 2. This gives us the flexibility of generating u_{i+1} weighted nodes from integral number of u_i weighted nodes, thereby eliminating splitting of demands.

In what follows, we present a sequence of lemmas, which help in guaranteeing the indivisibility constraint. Recall that Lemma 1 redistributes the weights uniformly at random and the probability that a vertex receives a weight of U is proportional to its weight.

Lemma 5. *Either there exists at least one arc with zero flow in the directed tour t constructed in procedure of of Lemma 1, or there exists a redistribution (using Lemma 1), with zero flow on at least one arc of the directed tour, which produces the exact same assignment of weights.*

Proof. The proof is complete if the first part of the lemma were true. Suppose it were not true. Let t be the directed tour in the procedure of Lemma 1, which was used to redistribute the weights. Let $m > 0$ be the smallest flow across a directed edge in t. Note that $m \leq U$. For each directed edge in t, subtract m from the flow on that edge. After this, we are guaranteed that at least one edge in t has a zero flow. Since this post-processing does not alter the distribution of weights, the proof is complete. ∎

Lemma 6. *There exists a redistribution using the procedure of Lemma 1 with $Y = U$, which produces the exact same assignment of weights as that with Y that is chosen uniformly at random from $(0, U]$.*

Proof. Let t be the directed tour in the proof of Lemma 1. As per Lemma 5, there exists at least one edge in t with zero flow. Let e be an edge in t from vertex p to vertex q with zero flow. Without loss of generality, we can assume that $p \in D$. As per the construction in the proof of Lemma 1, p must be one of the vertices that must have been marked. Since the flow on e is zero, it must be that Q_{new} at p is equal to $xU + Y$ for some integer x, which means that vertex g marked just after p must either have $(x+1)U + Y \in (Q_{old}$ at g, Q_{new} at $g]$ or $Y \in (Q_{old}$ at g, Q_{new} at $g]$. This means that Q_{new} at g is at least U greater than Q_{new} at p.

Recall from the proof of Lemma 1 that the vertices in t are visited starting from r. We now show that a construction with $Y = U$ on t, visiting vertices starting from q (instead of r) produces the exact same assignment of weights as that with Y that is chosen uniformly at random from $(0, U]$. From the above discussion, since Q_{new} at g is at least U greater than Q_{new} at p, and the flow on e is zero, it can be seen that the construction with $Y = U$ on t and visiting vertices starting from q produces the exact same outcome as what is desired, i.e., the set of vertices that were assigned a weight of U will exactly be the same as that marked in the proof of Lemma 1. ∎

The following lemma is easier than it appears, and differs from Lemma 1 in the following two aspects: (i) weights of vertices in T are powers of 2, and (ii) demand from a vertex is not split.

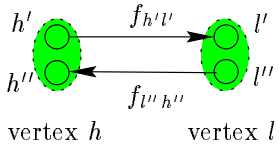

Fig. 1. l is a leaf node with h being its parent in T.

Lemma 7. *Let T be a tree with each edge having capacity U, a power of two. For all v in tree T, let $w(v)$ be a power of 2 with $w(v) < U$. Then there is an efficiently computable flow on T that redistributes the weights, respecting the cable capacity and without splitting a vertex weight, so that each vertex receives a new weight $w'(j)$ that is either 0 or U. Moreover,*

$$\mathbf{Pr}\ [\ \ w'(j) = U\ ] = w(j)/U$$

Proof. Using the argument in Lemma 6, we find a starting vertex from which we start visiting the vertices in the directed tour (obtained by replacing each edge in T with two oppositely directed arcs) in clockwise direction with $Y = U$. The value of Q is set to 0 initially. Increment Q by $w(j)$ on visiting vertex j. Let Q_{old} and Q_{new} be the value of Q just before and after visiting a vertex, respectively. Also, maintain set Z which is initially empty. Add v_j to Z on visiting vertex v_j. On visiting j, if for some integer x, $xU \in (Q_{old}, Q_{new}]$, then we do the following: (i) we find $W \subseteq Z$ such that $Q_{new} - xU = \sum_{i \in W} w(i)$, and (ii) ask the vertices in $Z \backslash W$ to send their weights to j while removing them from Z.

We now show how to find $W \subseteq Z$. Let g be the first vertex at which $Q_{new} \geq U$. The proof of Lemma 6 would have marked g and asked g to send $Q_{new} - U$ to the next marked vertex lying clockwise on the tour. We show that there exists a $W \subseteq Z$ whose removal from Z makes $Q_{new} - \sum_{i \in W} w(i) = U$. This is same as showing that there exists a set $M \subseteq Z$ such that $\sum_{i \in M} w(i) = U$. Recall that no vertex in Z has a weight more than U. To show that there exists an M, all we need to do is the following. Merge two vertices a and b of same weight w in Z into one vertex with weight $2w$. Since w is a power of 2, the weight of the new vertex remains a power of 2. Continue this merging process until (i) a vertex in Z is of weight U or (ii) no more merging is possible. While the former proves our claim, the latter is not possible as it is a contradiction to $\sum_{i \in Z} w(i) \geq U$, because $\sum_{k=0}^{i} 2^k < 2^{i+1}$. Once M is found, $W = Z \backslash M$. The vertices in W will be the sole contributors of the flow from g to the next vertex lying clockwise on the tour. This argument holds true for every vertex j at which $xU \in (Q_{old}, Q_{new}]$ for some integer x. Notice that the probability that a vertex $j \in T$ receives (gets assigned) a weight of U is $w(j)/T$, which is exactly what we needed, as per the lemma statement.

The proof will be complete once we show that the redistribution can be done on T rather than on the directed arcs of the Euler tour on T. Consider a leaf node $l \in T$ that is in the Euler tour. Let $h \in T$ be the node that was visited just before and after l (h is l's parent in T, which is rooted at r). We use x' and

x'' to represent vertex $x \in T$ in the directed Euler tour, with the tour entering x' and leaving x''. Let $f_{h'l'}$ and $f_{l''h''}$ be the flows on arcs from h' to l' and l'' to h'', respectively (see Fig. 1). During the redistribution process, if l had sent all its weight to some vertex— lying clockwise on the tour—that was assigned a weight of U, then ask h' to send the flow $f_{h'l'}$ directly to h'' instead of sending it through l. If l was assigned a weight of U in our redistribution process, then ask the vertices in W to reroute their flow bypassing l, i.e., make the flow coming into h' go directly to h'' instead of routing it through l. Remove l from T, and repeat this process for all leaf nodes in T. Note that whenever a leaf node is removed from T, the flow on the tree edge connecting that node to T is at most U. This process stops when there is just one node left in T. This completes the proof of Lemma 7. ∎

Let $G = (V, E)$ be the input graph with root $r \in V$, and let $D \subseteq V$ be the set of demands with d_j denoting the weight at j. Recall that the vertices in D may have non-integer weights because of the scaling we did to ensure $u_1 = \sigma_1 = 1$. Construct T_0, a ρ-approximate Steiner tree spanning D, using cables of capacity u_1. Use the procedure in the proof of Lemma 1 on T_0 with $U = u_1$, with $w(j)$ being the fractional part of $d_j \in D$, to collect demands at some subset of vertices in D such that the new weight $w'(j)$ of a vertex in D is either 0 or U. The cost of the redistribution will just be the cost of T_0. Since an optimal solution contains a candidate Steiner tree, the cost of T_0 is at most $\rho \sum_i C^*(i)/\sigma_i$.

As per the redistribution procedure, notice that (i) weight $w(j)$ of vertex $j \in T_0$ may have been split and assigned to at most two different nodes in T_0, and (ii) the weight of U is collected at a vertex $j \in D$ if and only if $w(j) > 0$, as the probability of a vertex getting assigned a weight of U is $w(j)/U$ (by Lemma 1). The former point is not consistent with our objective of routing the demands without having to split them across two nodes. To overcome the splitting, we round the integral demands at D up to the nearest powers of 2, and solve the problem for these new (rounded) weights. Even though, this means that we might install at most twice the required cable capacities, thereby losing a factor of 2 in the approximation ratio, we will have enough cable capacities installed so as route the original demands without having to split them.

Now, replace vertex $v \in D$ of weight $w(v)$ by $w(v)$ unit weight vertices. Let $\{v_1, \ldots, v_{w(v)}\}$ be the set of unit weight vertices that represent v. We call v to be the *origin* of v_i, $i = 1$ to $w(v)$. Our algorithm will ensure that the unit weight demands having a common origin travel together—along a single path—towards the sink.

The algorithm given below proceeds in the same manner as that in [5]. At the beginning of stage 1, $D_1 = D$ with each demand $j \in D$ having weight $d_j = 1 = u_1$. In general, at the beginning of stage t, D_t is a set of $|D|/u_t$ vertices, each with demand u_t. During stage t, our algorithm (presented below) uses the value u_{t+1} as the "aggregation threshold" to combine several demands of weight u_t into a single demand of weight u_{t+1}. The cables required to perform such an aggregation are bought by the algorithm. The demand will reach the root at the end of the algorithm. The final solution is then given by the union of

all the paths used in the aggregation stages. Given below are the steps performed at state t of the algorithm.

S1. *Mark* each demand v in D_t with probability $p_t = u_t/g_t$, and let D_t^* be the marked demands.

S2. Construct a ρ-approximate Steiner tree T_t on $F_t = D_t^* \cup \{r\}$. Install a cable of type $t+1$ on each edge of this tree.

S3. For each vertex $j \in D_t$, send its weight $w(j)$ to the nearest member of F_t using cables of type t. If two vertices have a common origin, ensure that both vertices send their weight to the same $i \in F_t$, as this guarantees that vertices having a common origin travel together, thus satisfying the indivisibility constraint. Let S_v be the set of vertices, with common origin v, that sent their weights to $i \in F_t$. Let $w_t(i)$ be the weight collected at $i \in F_t$.

S4. For each $i \in F_t$, order the vertices that sent their weight of u_t to i in such a way that the vertices in S_v are ordered before the vertices in S_u, if $|S_v| \geq |S_u|$.
Divide the vertices in the ordered set into groups of u_{t+1}/u_t vertices, starting from the first vertex, leaving behind $b_i = (\frac{w_t(i)}{u_t} \bmod \frac{u_{t+1}}{u_t})$ *residual* vertices at the end. Send back the weight of u_{t+1} emanating from each group of u_{t+1}/u_t vertices back from i to a random member of that group, buying new cables of type $t+1$. Since u_t, u_{t+1} and $|S_k|$, for all k, are powers of 2 by definition, our construction ensures the following: (i) set S_k, with $|S_k| \geq u_{t+1}/u_t$, is divided into p groups, where $p \geq 1$ is a power of 2, (ii) set S_k, with $|S_k| < u_{t+1}/u_t$ belongs to exactly one group. This implies that vertices with common origin travel together.

S5. For each $i \in F_t$, divide the b_i residual vertices into q_i sets $R_i^1, \ldots, R_i^{q_i}$, with each set containing vertices having common origin, and the weight $w(R_i^j)$ of a set R_i^j being the number of vertices it contains. Let $F_t' = \phi$ initially. For each $i \in F_t$, if $q_i \geq 1$, then add q_i copies of i into F_t', one for each set, with each copy carrying a weight of the sum of the vertex weights in the set that it represents. Observe that the weights of the vertices in F_t' are powers of 2. Also, note that T_t spans all the vertices in F_t', as the vertices in F_t' are mere copies of the vertices in F_t. Use the procedure of Lemma 7 on T_t spanning F_t' with $U = u_{t+1}$ to aggregate residual weights into groups of weight exactly u_{t+1} in a subset of vertices in F_t'. During the redistribution procedure, for every $i \in F_t$, ensure that its copies in F_t' are visited consecutively. This, along with the fact that $u_t \sum_{j=1}^{q_i} w(R_i^j) < U_{t+1}$ for every $i \in F_t$ ensures that at most one copy of i in F_t', representing $i \in F_t$, gets assigned a weight of u_{t+1}. Transform the redistribution among the vertices in F_t' into a redistribution among the vertices in F_t by assigning a weight of u_{t+1} to vertex $i \in F_t$ if one of i's copy was assigned a weight of u_{t+1} in F_t', and 0 otherwise. Notice that the probability that a vertex $i \in F_t$ is assigned a weight of u_{t+1} still depends on i's weight (residual weight, which is $b_i u_t$). For every $i \in F_t$, that receives a weight of u_{t+1}, choose a vertex $v \in b_i$ uniformly at random, and send the weight of u_{t+1} from i to v using cables of type $t+1$.

When $t = K$, we set the probability for non-root vertices $p_K = 0$, which implies that no vertex in stage $t = K$ is marked. The weights from all the vertices in D_K are directly routed to r using cables of capacity K. The approximation analysis for our SSBB algorithm is exactly the same as that for the Gupta et al.'s DSSBB algorithm [5]. All the lemmas used to prove Theorem 1 hold for our SSBB algorithm as well, but with $\epsilon = 1$. Recall that after the preprocessing step, we lose a factor of 2 from rounding up the weights of vertices to the nearest powers of 2. This means that our algorithm for the SSBB problem guarantees a ratio of twice that of Gupta et al.'s DSSBB algorithm. The cost C of our final solution is 2 times the cost in equation (3), and is given by

$$C < 2 \times 4 \times (2 + \rho + 1) \times 2(1 + 1) \sum_s C^*(s).$$

Using the current best approximation ratio of $\rho = 1 + \ln(3)/2$ [7] for finding a Steiner tree, we obtain a ratio of 145.6.

Theorem 2. *Our algorithm for the* SSBB *problem guarantees an approximation ratio of 145.6.*

References

1. B. Awerbuch and Y. Azar, "Buy-at-bulk network design," in *Proc. 38th IEEE Symp. on Foundations of Computer Science* (FOCS), pp. 542-547, 1997.
2. Y. Bartal, "Competitive analysis of distributed on-line problems-distributed paging," Ph.D. Dissertation, Tel-Aviv University, Israel, 1994.
3. 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. 8th Intl. Conf. on Integer Programming and Combinatorial Optimization* (IPCO), pp. 170-184, 2001.
4. S. Guha, A. Meyerson and K. Munagala, "A constant factor approximation for the single sink edge installation problems," in *Proc. 33rd ACM Symp. on Theory of Computing* (STOC), pp. 383-399, 2001.
5. A. Gupta, A. Kumar and T. Roughgarden, "Simpler and better approximation algorithms for network design," in *Proc. 35th ACM Symp. on Theory of Computing* (STOC), pp. 365-372, 2003.
6. R. Hassin, R. Ravi and F.S. Salman, "Approximation algorithms for capacitated network design problems," in *Proc. 3rd Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems* (APPROX), pp. 167-176, 2000.
7. G. Robins and A. Zelikovsky, "Improved Steiner tree approximation in graphs," in *Proc. 11th ACM-SIAM Symp. on Discrete Algorithms* (SODA), pp. 770-779, 2000.
8. F.S. Salman, J. Cheriyan, R. Ravi and S. Subramanian, "Approximating the single-sink link-installation problem in network design," *SIAM J. on Optimization*, 11(3), pp. 595-610, 2000.
9. K. Talwar, "Single-sink buy-at-bulk LP has constant integrality gap," in *Proc. 9th Conf. on Integer Programming and Combinatorial Optimization* (IPCO), pp. 475-486, 2002.