Multicast Routing for Energy Minimization Using Speed Scaling

Nikhil Bansal[1,⋆], Anupam Gupta[2,⋆⋆], Ravishankar Krishnaswamy[3,⋆⋆⋆], Viswanath Nagarajan[4], Kirk Pruhs[5,†], and Cliff Stein[6,‡]

[1] Mathematics and Computer Science, Eindhoven University of Technology
[2] Computer Science, Carnegie Mellon University
[3] Computer Science, Princeton University
[4] IBM T.J. Watson Research Center
[5] Computer Science, University of Pittsburgh
[6] Industrial Engineering and Operations Research, Columbia University

Abstract. We consider virtual circuit multicast routing in a network of links that are speed scalable. We assume that a link with load f uses power $\sigma + f^{\alpha}$, where σ is the static power, and $\alpha > 1$ is some constant. We assume that a link may be shutdown if not in use. In response to the arrival of client i at vertex t_i a routing path (the virtual circuit) P_i connecting a fixed source s to sink t_i must be established. The objective is to minimize the aggregate power used by all links.

We give a polylog-competitive online algorithm, and a polynomial-time $O(\alpha)$-approximation offline algorithm if the power functions of all links are the same. If each link can have a different power function, we show that the problem is APX-hard. If additionally, the edges may be directed, then we show that no poly-log approximation is possible in polynomial time under standard complexity assumptions. These are the first results on multicast routing in speed scalable networks in the algorithmic literature.

1 Introduction

The amount of energy used by data networks is significant, worldwide more 50 billion kWH are used per year according to a US Department of Energy study [1]. As the number of processors per chip grows, interprocessor communication is widely expected to become the dominant energy component for computer chips. Thus there has been significant recent interest, both within industry

⋆ Supported in part by the NWO Vidi grant 639.022.211.
⋆⋆ Supported in part by NSF awards CCF-0964474 and CCF-1016799.
⋆⋆⋆ Supported in part by NSF awards CCF-0964474 and CCF-1016799 and an IBM Graduate Fellowship.
† Supported in part by NSF grants CCF-0830558, CCF-1115575, CNS-1253218 and an IBM Faculty Award.
‡ Supported in part by NSF grant CCF-0915681.

and academia, to develop methods to manage networks, from on-chip up to wide-area, in a more energy efficient manner. The same US Department of Energy study [1] estimates that at least a 40% reduction in wide-area network energy would be possible if network components were able to dynamically adjust their power in response to the traffic experienced. Simulation results by Benoit et al. [5] suggest using speed scalable link technology, such as those proposed by Kim and Horowitz [11], would save significant energy in an on-chip network for a chip multiprocessor.

Virtual circuit routing is a common means of providing reliable communications in a network. A virtual circuit gives the user a reserved portion of the network with a guaranteed bandwidth in which to route messages. Algorithmic problems associated with virtual circuit routing have been studied for many years (e.g. [4,7]). Here we consider algorithmic problems that combine the traditional virtual circuit routing with the the traditional energy management mechanisms: changing the speed of a component and/or shutting the component off.

We consider the same setting as was proposed in several previous papers. The network is a graph, with a power function associated with each edge.[1] The power used by edge e as

$$\mathrm{en}_e(f_e) = \begin{cases} \sigma_e + f_e^\alpha & \text{if } f_e > 0 \\ 0 & \text{if } f_e = 0 \end{cases}, \tag{1}$$

where f_e is the number of circuits (flow) passing through each edge, σ is the static power, and $\alpha > 0$ is a given constant. An important special case is when all edges have the same fixed cost σ. We call such cost functions *homogeneous* and call the general case *heterogeneous*. We consider both directed and undirected graphs, unless explicitly stated graphs are undirected. The input further consists of a sequence of requests for the establishment of a virtual circuit. In response to the arrival of a source-sink request (s_i, t_i) a routing path (the virtual circuit) P_i connecting s_i to t_i must be established. The objective is to minimize the aggregate power used over all the edges.

Andrews et al. [2] gave a polynomial-time poly-log-approximation algorithm if the graph is undirected, and edges e have homogeneous cost functions. It was previously noted [3] that a polynomial-time algorithm with a better than poly-log approximation ratio would violate standard complexity theoretic assumptions. Gupta et al. [8] considered online algorithms and gave an α^α-competitive online algorithm under the same assumptions, plus the additional assumption that the static power σ_e for each edge was zero.

In this paper we consider applications in which there is a common source vertex, as would be the case if a multicast communication pattern was implemented using unicast. We consider both the online and offline cases and give several positive and negative results:

- For undirected networks and homogeneous power functions of the form given in (1), in Section 3 we give a poly-log competitive online algorithm.

[1] Power functions are associated with edges and not vertices primarily because the resulting algorithmic problems are more tractable.

(These are the same assumptions under which an offline poly-log approximation algorithm was given by [2], although the result of [2] holds more generally without the single source assumption.) This result shows that poly-log competitiveness can be achieved by an online algorithm for multicast communication.
– For undirected networks and homogeneous power functions of the form given in (1), in Section 4, we give an $O(\alpha)$ offline approximation algorithm.
– In Section 5.1, we show that if the graph is directed instead of undirected, and we allow heterogeneous power functions, then even for the s–t routing case (i.e. when all the sinks are also located at the same node), there is no polynomial-time poly-log-approximation algorithm under a standard complexity theoretic assumption.
– In Section 5.2, we show that for the heterogeneous case in an undirected graph, the offline problem is APX-hard even in the s–t case.

In Section 6 we discuss some of the natural open questions arising from these results.

2 Notation and Background

In the Energy-Aware Routing Problem (EERP), we are given an undirected graph $G = (V, E)$ with $|V| = n$ and a distinguished source vertex s. We are given k sinks $R_k = \{t_1, t_2, \ldots, t_k\}$ corresponding to k different routing requests, One unit of flow needs to be routed from the source s to each sink t_i, on a single s-t_i path P_i. Given a solution $P_1, P_2, \ldots, P_k$, the flow on edge e, f_e, is the number of paths that use e, that is $f_e = \sum_{i:e\in P_i} 1$, and the energy used by edge e is defined by equation (1) above. For our positive results, we consider the homogeneous case where all the edges have a common σ. Furthermore, we are interested in the case where $1 \ll \sigma$ and $\alpha > 1$. (If σ is small, for the purposes of approximation, it can be treated as 0, and if $\alpha < 1$, this problem exhibits economies of scale and has been well studied.) The objective is to find a feasible routing which minimizes the total power, which is obtained by summing the power usage over all edges. In other words, the total power used is

$$\sum_{e\in E} \mathrm{en}_e = \sum_{e\in E} \left(\sigma \cdot \mathbf{1}_{(e\in\cup_i P_i)} + \left(\sum_{i=1}^{k} \mathbf{1}_{(e\in P_i)} \right)^{\alpha} \right) \tag{2}$$

We will usually refer to the power used as the *cost incurred*.

We consider both the offline and online variants: in the online variant, the sinks are given online and we must choose a path for sink t_i before learning about sink t_{i+1}.

The fixed σ term is called the *buying* or *opening* cost: the net buying cost is the first term in (2). The second term is sometimes called the *renting* cost. In designing our algorithms, we will sometimes want to balance the renting and buying costs, we therefore define $q = \sigma^{1/\alpha}$, the minimum number of paths that must use an edge for the renting cost to be at least the buying cost.

We will compare our results to an optimal solution and use $\mathsf{opt}(R_k)$ to denote the cost incurred by the set of optimal paths $\{P_1^*, P_2^*, \ldots, P_k^*\}$.

Steiner Trees: We will use Steiner trees, both in our algorithm design and for showing lower bounds. Given $S \subseteq V$, a *min-cost Steiner tree on* S is a tree T in G with the fewest edges that connects all the nodes in S. We denote the number of edges in the min-cost Steiner tree by $\mathsf{StTree}(S)$. Since the union of the paths P_i in any solution $\{P_i\}_{i=1}^k$ to EERP also connects $\{s\} \cup R_k$, we get that the optimal buying cost is at least $\sigma \cdot \mathsf{StTree}(\{s\} \cup R_k)$, and hence

$$\mathsf{opt}(R_k) \geq \sigma \cdot \mathsf{StTree}(\{s\} \cup R_k). \tag{3}$$

We will sometimes use opt as shorthand for $\mathsf{opt}(R_k)$.

Probabilistic Bounds: Our analysis will use the following result on the expectations of the sums of powers of random variables.

Theorem 1 ([10,13]). *Let $X_1, X_2, \ldots, X_N$ be independent non-negative random variables. Let $\alpha > 1$ and $K_\alpha = \Theta(\alpha/\log\alpha)$. Then it is the case that* $(\mathbb{E}[(\sum_i X_i)^\alpha])^{1/\alpha} \leq K_\alpha \ \max\left(\sum_i \mathbb{E}[X_i], \left(\sum_i \mathbb{E}[X_i^\alpha]\right)^{1/\alpha}\right)$.

Corollary 1. *Let $p \geq 0$, and let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables taking value D with probability $\max\{1, p\}$. Then $\mathbb{E}[(\sum_i X_i)^\alpha] \leq (K_\alpha)^\alpha \cdot \max\{1, pN\, D^\alpha + (pND)^\alpha\}$, where $K_\alpha = \Theta(\alpha/\log\alpha)$.*

Proof. For the case when $p \geq 1$, $X_i = D$ with probability 1, and hence we can conclude that $\mathbb{E}[(\sum_i X_i)^\alpha] = (ND)^\alpha$. For the case when $p \in [0,1]$, $\mathbb{E}[X_i] = pD$, and $\mathbb{E}[X_i^\alpha] = pD^\alpha$. From this we can conclude that the upper bound in Theorem 1 is $K_\alpha \max(pND, (pN)^{1/\alpha} D)$. Taking α^{th} powers and replacing the max by a sum, we get $\mathbb{E}[(\sum_i X_i)^\alpha] \leq (K_\alpha\, D)^\alpha ((pN)^\alpha + pN)$. $\square$

3 Online Algorithm for Homogeneous Setting

We now give an online algorithm, in which the sinks arrive one-by-one, and we have to choose the path P_i connecting t_i to the source s before knowing the identity of the next sink t_{i+1}.

3.1 The Algorithm

We assume (without loss of generality, by adding zero cost edges) that each sink appears at a distinct vertex; so the number of sinks $k \leq n$ the number of vertices in G. We let $R_i = \{t_1, t_2, \ldots, t_i\}$ denote the set of demands that have already arrived and let $R_i' = \{s\} \cup R_i$. The algorithm maintains a tree T_i that initially contains only the source s. When a request t_i arrives, a path P_i will be chosen and the tree will be updated, however, the routing paths may use edges outside this tree. We will also maintain a subset of the vertices which we call *leaders*. The set of leaders in step i is denoted by L_i, and is initialized to $L_0 = \{s\}$. We will maintain various counters, all initialized to 0. A counter Λ_j will denote the number of nodes assigned to a particular leader j. A counter ρ_e will count the number of times an edge has been used in non-tree paths from the source to a leader and a counter λ_e will count the number of times an tree-edge is used to route flow.

We will denote trees and paths by their sets of edges.

When a sink t_i arrives, we do the following:

1. Let $\tilde{P}_i$ be the shortest path from t_i to any node in R'_{i-1} and let $T_i = T_{i-1} \cup \tilde{P}_i$.
2. With probability $\min\{1, \frac{c \log n}{q}\}$, let t_i be a leader (i.e., set $L_i = L_{i-1} \cup \{t_i\}$), else let $L_i = L_{i-1}$.
   If t_i is a leader, find a path Q_i from t_i to the root that minimizes
   $$\sum_{e \in Q_i} \left((\rho_e + 1)^\alpha - (\rho_e)^\alpha \right). \tag{4}$$
   For each edge e on the path Q_i, set $\rho_e \leftarrow \rho_e + 1$.
3. Choose the leader $j \in L_i$ that minimizes the expression:
   $$\begin{aligned} &(3/2)^{(\Lambda_j+1)/(q \log n)} - (3/2)^{\Lambda_j/(q \log n)} \\ &+ \sum_{e \in T_i[t_i, j]} \left((3/2)^{(\lambda_e+1)/(q \log n)} - (3/2)^{\lambda_e/(q \log n)} \right) \end{aligned} \tag{5}$$
   where $T_i[a, b]$ is the unique path between nodes a and b in the tree T_i. If this minimizer is the leader $j^* \in L_i$, set $\Lambda_{j^*} \leftarrow \Lambda_{j^*} + 1$; for every edge e in $T_i[t_i, j^*]$, set $\lambda_e \leftarrow \lambda_e + 1$.
4. Set the s-t_i path P_i to be Q_{j^*} (which is a s-j^* path) concatenated with $T_i[j^*, t_i]$. We call the latter part of the path P_i to be the "tree" portion, and the former to be the "non-tree" portion.

Above, $c > 1$ is some constant. The choice of expression (5) is from the online path selection algorithm to minimize the congestion (maximum load) [4]; similarly the choice of (4) is from the online algorithm to minimize the sum of α^{th} powers of edge loads [8].

3.2 Analysis

The algorithm maintains the tree T_i, the source to leader paths Q_{j^*} and the routing paths P_i. Recall that the flow on an edge e is defined from the P_i. Observe that it is an easy consequence of the algorithm that after step k' there is non-zero flow on all edges in $T_{k'} \cup \cup_{j^* \in L_{k'}} Q_{j^*}$, and hence we have "bought" all of these edges. In the following analysis, we will use k as an index into the number of sinks, rather than (necessarily) the total number of sinks. We first bound the cost associated with the tree-edges. We divide the analysis into buying cost (the number of tree edges) and renting cost (a function of the load on the tree edges).

Claim 1. *The total buying cost for edges in T_k is at most $O(\log k)\mathsf{opt}(R_k)$.*

Proof. The edges in T_k are all bought in Step 1. This step implements a greedy Steiner tree algorithm, and hence the number of edges bought by the greedy Steiner tree algorithm is at most $O(\log k)$ times the optimal Steiner tree on R'_k [9]. As mentioned in Section 2 the optimal Steiner tree gives a lower bound on $\mathsf{opt}(R_k)$, and we get that the total cost of edges bought in Step 1 is at most $O(\log k)\mathsf{opt}$. □

Now we show that in step 3, no edge is used too many times.

Lemma 1. *With probability at least $1 - n^{-2\alpha}$ over the choice of the leaders:*

1. *There exists an assignment of sinks to leaders so that (a) each edge of T_k is used in the tree portion of at most $O(q \log n)$ sinks, and (b) each leader is assigned at most $O(q)$ sinks.*
2. *Thus our algorithm obtains a path assignment in Step 3 where each edge of T_k is used $O(q \log^2 n)$ times and each leader is assigned $O(q \log n)$ sinks.*

Proof. Each sink becomes a leader with probability $\min\{1, \frac{c \log n}{q}\}$. So if $q \leq c \log n$ then no edges are ever used and each leader is only assigned one sink (namely itself). For the rest of the argument we assume that $q \geq c \log n$.

For the sake of the analysis, we choose the leaders in a different way: for each sink, we flip $c \log n$ independent coins of bias $1/q$, if the i^{th} coin is the first of these to turn up heads, we designate the sink as a leader of color i. If all coins turn up tails, the sink is not considered a leader. Since the probability that a sink is a leader (of any color) according to this process is $1 - (1 - 1/q)^{c \log n} \leq \frac{c \log n}{q}$, proving the statements with this new way of choosing leaders also proves the original statement (via a standard coupling argument).

Consider the tree T_k and partition it into connected edge-disjoint groups, so that each group (apart from possibly one) contains between $2q$ and $4q$ sinks. This partition can be achieved by rooting the tree T_k, making it binary by adding dummy edges, and repeatedly choosing the deepest subtree containing at least $2q$ sinks. Consider a particular group S of sinks, and order the sinks in S according to their arrival times. The probability that there is at least one leader of color 1 among the first q sinks in S is at least $1 - (1 - 1/q)^q \geq 1/2$. For such groups, at least half their sinks can route to this leader of color 1. Since the groups were chosen to included edge-disjoint parts of the tree, this routing incurs a maximum load of $4q$ on any edge of the tree (and on any color-1 leader). Also, this reduces the number of remaining sinks to $3k/4$ in expectation. Now we can recurse on the remaining sinks: form edge-disjoint groups on them, and assign $3/4$ fraction of these sinks (in expectation) to leaders of color 2, and continue. After repeating this process $c \log n$ times, the expected number of unassigned sinks is at most $(3/4)^{c \log n} \cdot k \leq \frac{k}{n^{3\alpha}} \leq \frac{n}{n^{3\alpha}}$, by setting $c \geq 9\alpha$. Thus (by Markov's inequality) there is no unassigned sink with probability at least $1 - n^{-2\alpha}$.

Altogether, this assignment routes at most $4q$ sinks to each leader, and uses each edge of T_k at most $4c \log n \cdot q$ times. This proves the first part of the lemma.

If each edge of T_k is given capacity $O(q \log n)$ and each leader is given capacity $O(q)$, the above path assignment corresponds to a solution having congestion (i.e. load/capacity) one for the following routing problem: each sink t_i has to use edges of T_k to route unit flow from any node of L_i (i.e. leader among $\{t_1, \dots, t_i\}$). The result of Aspnes et al. [4] implies that path assignment according to (5) gives a solution with congestion $O(\log n)$ times the optimal. This proves the second part of the lemma. $\square$

We now combine the previous two arguments.

Lemma 2. *The expected total cost (from both buying and renting) incurred by the algorithm on the tree portions of the paths* $\{P_i\}_{i=1}^k$ *is* $O(\log^{2\alpha} n \log k)\mathsf{opt}(R_k)$.

Proof. By Claim 1, the buying cost for the tree edges is $O(\log k)\mathsf{opt}(R_k)$, and hence the number of edges bought $|T_k| \le O(\log k)\mathsf{opt}(R_k)/\sigma$.

By the second part of Lemma 1, with probability at least $1 - n^{-2\alpha}$, each edge of T_k carries at most $O(q \log^2 n)$ flow. Thus the expected renting cost incurred over T_k is at most $|T_k| \cdot O(q \log^2 n)^\alpha + \frac{1}{n^{2\alpha}} |T_k| \cdot k^\alpha \le O(\log n)^{2\alpha} |T_k| \cdot \sigma = O(\log k \cdot \log^{2\alpha} n)\mathsf{opt}(R_k)$. $\square$

We now bound the cost of using the edges in the non-tree portion $\cup Q_{j*}$.

Lemma 3. *Consider the following random experiment: choose a random subset* S *of sinks, with each sink* t_i *chosen independently with probability* $\min\{1, \frac{c \log n}{q}\}$*; thereafter for each* $t_i \in S$*, send* $\Theta(q \log n)$ *flow from* s *to* t_i *on its optimal path* P_i^**. The expected cost (both buying and renting) incurred by this routing is* $O(\log^{2\alpha} n)\mathsf{opt}(R_k)$.

Proof. Since we buy a subset of the edges bought by the optimal solution, the buying cost is bounded by $\mathsf{opt}(R_k)$. For the expected renting cost, consider an edge e, and all the sinks whose optimal paths P_i^* use e: if there are N of them, the optimal's renting cost for e is N^α. Since each sink chooses independently, we can use Corollary 1 with $p = \frac{c \log n}{q}$ and $D = c' q \log n$ to bound the expected renting cost for e. Ignoring terms of the form $O(\alpha)^\alpha$ and using $q^\alpha = \sigma$, we get

$$pND^\alpha + (pND)^\alpha \approx (\log^{\alpha+1} n)\, \sigma N/q + (\log^{2\alpha} n)\, N^\alpha \le (\log^{2\alpha} n)(2N^\alpha + \sigma),$$

which is the claimed polylogarithmic factor times the optimal's cost incurred on this edge. (For the last inequality, observe that if $q \le N$ then $\sigma N/q = q^{\alpha-1} N \le N^\alpha$, and if $q > N$ then $\sigma N/q \le \sigma$.) Now summing over all edges, and using linearity of expectations completes the proof. $\square$

Lemma 4. *The expected cost incurred by our algorithm for routing on the non-tree edges* $\cup_{j^*} Q_{j*}$ *is at most* $O(\log^{2\alpha} n)\mathsf{opt}(R_k)$.

Proof. Consider a random instance on the original graph where each sink is activated independently (as leader) with probability $\frac{c \log n}{q}$ and requires $\Theta(q \log n)$ unsplittable flow from s, with the objective of minimizing $\sum_{e: f_e > 0} (\sigma + f_e^\alpha)$ where

f_e denotes the flow on edge e. Since the routing is unsplittable, each positive f_e has $f_e \geq \Omega(q \log n) \geq q$; so $\sigma + f_e^\alpha \leq 2 \cdot f_e^\alpha$. Thus (up to a factor of 2) the objective is simply $\sum_e f_e^\alpha$. By Lemma 3, the expected optimal value of this random instance is $O(\log^{2\alpha} n)\mathsf{opt}(R_k)$. The path selection $\{Q_j\}$ in Step 2 corresponds to an α^α-competitive online algorithm for this random instance [8]. Thus, if we send $O(q \log n)$ flow along each of these paths, the expected cost incurred is $\alpha^\alpha \cdot O(\log^{2\alpha} n)\mathsf{opt}(R_k) = O(\log^{2\alpha} n)\mathsf{opt}(R_k)$.

Now, by the second part of Lemma 1, with probability at least $1 - n^{-2\alpha}$, the number of sinks assigned to each leader is $O(q \log n)$, in which case reserving capacity $O(q \log n)$ on each leader's path from s (as in above instance) suffices. With the remaining $n^{-2\alpha}$ probability, the worst case cost is $nk(\sigma + k^\alpha)$. Thus the expected cost of our algorithm on the non-tree portion is $O(\log^{2\alpha} n)\mathsf{opt}(R_k)$. □

Theorem 2. *There is an $O_\alpha\left(\log^{O(\alpha)} n\right)$-competitive randomized online algorithm for single-source* EERP *with homogeneous power functions of form $\sigma + f_e^\alpha$.*

4 Offline Algorithm for Homogeneous Setting

In this section we give an $O(\alpha)$-approximation algorithm for the *offline* EERP problem. with a homogeneous energy function. The algorithm has two phases (similar to the online setting), aggregation and batched routing. We assume that $\sigma \geq \alpha^\alpha$; otherwise aggregation is not necessary and the algorithm proceeds directly to the network flow instance[2].

Set $p := \sigma^{1/\alpha}/\alpha \geq 1$, which may not be integral. We first describe an algorithm to compute a splittable routing for each sink: then we show that this can be easily converted to an unsplittable routing.

Aggregation: Let T denote an approximately minimum Steiner tree, which we can compute in polynomial time. Using an Euler tour of T we can fractionally partition the k demands to obtain r groups $\{V_j \subseteq \{t_1, \ldots, t_k\}\}_{j=1}^r$ where V_j induces a subtree T_j on T, so that:

- For each group $j \in [r]$, there is positive (fractional) demand only on sinks V_j, which totals to exactly p.
- For each sink t_i, the total demand over all groups is exactly one.
- Each edge of T appears in at most two subtrees $\{T_j\}_{j=1}^r$.

If k is not an integral multiple of p, by adding dummy demand, we can ensure that each group contains exactly p demand (this only affects the approximation ratio by a constant factor). In doing so, we may need to add one fractional demand, for that sink the second condition is modified so that the total demand is equal to the fractional amount.

[2] In this case the algorithm is even simpler: all capacities are one and the copies of an edge are: $\lceil \alpha \rceil$ edges of cost σ each; and for each integer $h \geq \lceil \alpha \rceil + 1$, an edge of cost $h^\alpha - (h-1)^\alpha$. That this is an $O(\alpha)$ approximation, follows easily from the arguments for "Batched Routing" below.

Batched routing: We now define a minimum-cost network flow instance G' corresponding to the above grouping $\{V_j\}_{j=1}^r$ of sinks. We create r new sinks $\{t'_j\}_{j=1}^r$ where each t'_j requires flow p and is connected to all sinks in V_j with zero cost and infinite capacity edges. Then we replace each edge e in G by the following parallel edges:

– There are $\lceil\alpha\rceil$ identical edges with capacity p and cost (per unit flow) $\frac{\sigma}{p}$.
– For each integer $h \geq 0$, there is an edge of capacity p and cost $\left(1+\frac{h}{\alpha}\right)^{\alpha-1}\cdot\frac{\sigma}{p}$.

We use $g(x) = \sigma + x^\alpha$ to denote the homogeneous power function applied to the flow on one edge. The transformation above corresponds to a natural discretization of the power function into linear pieces, and was also used in Andrews et al. [2]. Let $c_e(x)$ denote the minimum cost way to send x units of flow through the above set of parallel edges corresponding to an edge e. This minimum cost routing uses the edges in order of increasing cost.

Before analyzing our algorithm, we prove two technical claims about the behavior of our cost functions. The choice of our discretization parameter p implies:

Claim 2. *For all $x \geq 0$, $g(x+p) \leq 9 \cdot g(x)$.*

Moreover, by the definition of the parallel edges,

Claim 3. *For each $x \geq 0$, $c(x) \leq (\alpha+1)\cdot g(x)$; and for each $x \geq p$, $g(x) = O(c(x))$.*

We now return to the description of our algorithm. The algorithm computes a minimum cost flow in this network G' with demands of p units to each of $t'_1, \ldots, t'_r$. Since all capacities and demands are multiples of p, we can obtain in polynomial time (by integrality of single commodity flow) an optimal solution given by paths $\{Q_j\}_{j=1}^r$ where each Q_j is a path from s to some $t^*_j \in V_j$ carrying p flow. For each edge $e \in G$ (the original graph), let f_e denote the total flow sent through "copies" of e in this solution; note that since f_e is a multiple of p, either $f_e = 0$ or $f_e \geq p$. Let $E' \subseteq E$ denote the edges e with $f_e > 0$, so the cost of this solution is $\sum_{e\in E'} c(f_e)$.

Lemma 5. *The cost of the flow $\sum_{e\in E'} c(f_e) \leq (\alpha+1)\cdot\mathsf{opt}$. Moreover, the total energy cost, $\sum_{e\in E'} g(f_e) = O(\alpha)\cdot\mathsf{opt}$.*

Proof. We will show the existence of a feasible solution of cost $(\alpha+1)\cdot\mathsf{opt}$ to the above network flow instance: since we obtain an optimal solution, our cost $\sum_{e\in E'} c(f_e)$ is no worse. Consider the optimal paths $\{P^*_i\}_{i=1}^k$ in EERP carrying unit flow to each sink $\{t_i\}_{i=1}^k$. For each group $j \in [r]$ and sink $t \in V_j$ we send $\mathsf{demand}_j(t)$ units of flow to t'_j by extending path P^*_t. The property of the aggregation step ensures that each $\{t'_j\}_{j=1}^r$ receives exactly p flow, and the flow f^*_e through any edge in G is exactly the number of paths $\{P^*_i\}_{i=1}^k$ using it. Thus the cost of this solution is $\sum_e c(f^*_e) \;\leq_{Claim\ 3}\; (\alpha+1)\sum_e g(f^*_e) = (\alpha+1)\cdot\mathsf{opt}$.

Next, observe that our optimal solution $\{f_e : e \in E'\}$ has any positive flow at least p. Hence using Claim 3, $\sum_{e\in E'} g(f_e) \leq O(1)\sum_{e\in E'} c(f_e) = O(\alpha)\cdot\mathsf{opt}$. $\square$

Obtaining a solution. We now combine the solutions from the above two phases to obtain a *splittable* EERP routing in G. The flow $\{f_e : e \in E'\}$ sends p units of flow to $t_j^* \in V_j$ for each group $j \in [r]$. Then for each $j \in [r]$, using the edges on subtree T_j, these p units can be redistributed from t_j^* so that each sink $t \in V_j$ gets exactly $\mathsf{demand}_j(t)$ flow, and the flow on each edge of T_j is at most p. This flow is a feasible splittable EERP solution, since the total demand of each sink (over all groups) is one. Since each edge in T appears in at most two subtrees $\{T_j\}_{j=1}^r$, the final flow on any edge e is at most $f_e + 2p$. So the cost of this combined routing is at most:

$$\sum_{e\in E'} g(f_e + 2p) + \sum_{e\in T\setminus E'} (\sigma + (2p)^\alpha) \le 9^2 \cdot \sum_{e\in E'} g(f_e) + 9^2 \cdot \sum_{e\in T\setminus E'} \sigma$$
$$\le 9^2 \sum_{e\in E'} g(f_e) \;+\; 2 \cdot 9^2 \cdot \mathsf{opt}$$

The first inequality is by Claim 2 and the definition of p. The second inequality is by the fact that T can be chosen to be a 2-approximate Steiner tree. Finally, using Lemma 5, $\sum_{e\in E'} g(f_e) = O(\alpha)\cdot\mathsf{opt}$, and the total cost of this splittable routing is $O(\alpha)\cdot\mathsf{opt}$.

We now obtain an unsplittable routing. If $\{\ell_e\}_{e\in E}$ denotes the flow in this solution, define a network with each edge $e \in G$ having capacity $\lceil \ell_e \rceil$. The source is s and there is a unit demand at each $\{t_i\}_{i=1}^k$. This instance is feasible as shown by the fractional solution $\{\ell_e\}_{e\in E}$. Again, using integrality of flow, we can find an integer valued flow within these capacities, yielding the unsplittable EERP solution. Using Claim 2, we see that the cost is at most $\sum_e g(\ell_e + 1) \le \sum_e g(\ell_e + p) \;\le\; 9\sum_e g(\ell_e) = O(\alpha)\cdot\mathsf{opt}$.

Theorem 3. *There is an $O(\alpha)$-approximation algorithm for EERP with a single source and homogeneous power functions of the form $\sigma + f_e^\alpha$.*

5 Hardness of Approximation Results

5.1 Hardness of *s*-*t* Directed Routing with Heterogeneous Functions

We now consider the heterogeneous case in which all sinks are located at the same node. Since all sinks are located at the same node, we will speak of a demand r (which is equivalent to the number of sinks k in our original formulation). We therefore want to compute an s-t flow of r units having the minimum total power (summed over all edges). We will consider the case when the exponent α being a small constant that is larger than one, i.e. $\alpha = 1+\epsilon$ where $\epsilon > 0$ is any constant. The main result of this section is the following inapproximability:

Theorem 4. *The s-t routing problem on directed graphs with heterogeneous power functions and constant exponent $\alpha > 1$ does not admit an approximation ratio better than $2^{\log^{1-\delta} n}$ for any $\delta > 0$, unless $NP \subseteq DTIME(\text{polylog(n)})$.*

Proof. We reduce from the label cover problem. The input is a bipartite graph $(A \cup B, F)$ where each vertex in A and B has degree d, a label set L and a relation $\pi_{a,b} \subseteq L \times L$ for each $(a,b) \in F$. We let $|A| = |B| = n$ and $|F| = m = n \cdot d$. The goal is compute a labeling $\phi : A \cup B \to L$ that maximizes the number of consistent edges, where edge $(a,b) \in F$ is consistent if $(\phi(a), \phi(b)) \in \pi_{a,b}$. For any $\delta > 0$ it is known [12] that unless $NP \subseteq DTIME(\text{polylog(n)})$, there is no polynomial time algorithm to distinguish between:

– Yes instances: with optimal value $|F|$, i.e. there is a labeling with all edges consistent.
– No instances: with optimal value at most $|F|/2^{\log^{1-\delta} n}$, i.e. no labeling makes more than $|F|/2^{\log^{1-\delta} n}$ edges consistent.

The reduction here is similar to one for the related *s-t capacitated network design problem* [6], where each edge has a fixed cost and capacity (instead of a power function), and the goal is to choose a minimum-cost set of edges that support f units of flow from s to t. It is straightforward to reduce s-t capacitated network design to our problem when the exponent $\alpha \approx n$. Below we show that the same hardness persists even for any constant exponent $\alpha > 1$.

The graph $G = (V, E)$ for the heterogeneous power s-t directed routing problem is as follows. The vertex set $V = \{s,t\} \cup A \cup B \cup \{a(u) : a \in A, u \in L\} \cup \{b(u) : b \in B, u \in L\}$. The edge set E contains:

– For each $a \in A$, there is an edge (s, a) with function $d \cdot f^\alpha$. (To ensure functions of the form $\sigma_e + f^\alpha$, we can subdivide this edge into d smaller edges each having function f^α.)
– Similarly, for each $b \in B$, there is an edge (b, t) with function $d \cdot f^\alpha$.
– For each $a \in A$ and $u \in L$, there is an edge $(a, a(u))$ with function $d^{\alpha+1} + f^\alpha$.
– Similarly, for each $b \in B$ and $u \in L$, there is an edge $(b(u), b)$ with function $d^{\alpha+1} + f^\alpha$.
– For each $(a,b) \in F$ and $(u,v) \in \pi_{a,b}$ there is an edge $(a(u), b(v))$ with function $d^\alpha \cdot f^\alpha$. (Again each such edge can be subdivided into d^α edges with function f^α.)

We denote the last set of edges as E'. The flow demand is set to $m = |F| = d \cdot n$ units.

Yes instances: Observe that if the label cover instance has a labeling ϕ that is consistent for all edges, there is a routing in G of total cost at most $7m \cdot d^\alpha$.

No instances: Suppose that there is a routing in G of total power $\rho \cdot m \cdot d^\alpha$. Then we show that one can recover a labeling for the label cover instance that satisfies at least $|F|/\rho^{2+\frac{3}{\alpha-1}}$ edges.

Let f_e denote the flow on edge e in the given routing (having power $\rho \cdot m \cdot d^\alpha$). For each $a \in A$, define $N_a := \{u \in L : f_{(a,a(u))} > 0\}$. Similarly for $b \in B$, $N_b := \{u \in L : f_{(b(u),b)} > 0\}$. We consider an arbitrary flow decomposition of $\{f_e\}_{e \in E}$ into s-t flow-paths (of total value m), and modify it as follows (below $\beta := (4\rho)^{\frac{1}{\alpha-1}}$).

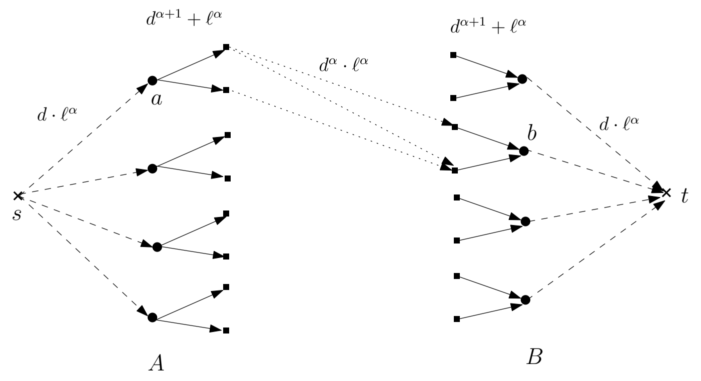
$d^{\alpha+1}+\ell^\alpha$
$d^{\alpha+1}+\ell^\alpha$
$d^\alpha \cdot \ell^\alpha$
$d \cdot \ell^\alpha$
$d \cdot \ell^\alpha$
a
b
s
t
A
B

Fig. 1. The $s-t$ directed network with power functions

1. For each $a \in A$, if $f_{(s,a)} > \beta \cdot d$ or $|N_a| > 4\beta\rho$ then delete all flow-paths through (s,a).
2. For each $b \in B$, if $f_{(b,t)} > \beta \cdot d$ or $|N_b| > 4\beta\rho$ then delete all flow-paths through (b,t).
3. For each $e \in E'$, if $f_e > \beta$ then delete all flow-paths through e.

Claim 4. *The total flow remaining after this pruning is at least* $\frac{m}{4}$.

Proof. We bound the flow lost in each step separately. Consider first the edges $E_a = \{(s,a) : f_{(s,a)} > \beta \cdot d\}$. We have:

$$\frac{\sum_{e\in E_a} f_e}{\sum_{e\in E_a} f_e^\alpha} \quad\le\quad \max_{e\in E_a} f_e^{1-\alpha} \quad\le\quad \frac{1}{(\beta d)^{\alpha-1}} \quad\Longrightarrow\quad \sum_{e\in E_a} f_e \quad\le\quad \frac{\sum_{e\in E_a} f_e^\alpha}{(\beta d)^{\alpha-1}}$$

An identical analysis implies that $\sum_{e\in E_b} f_e \le (\beta d)^{1-\alpha} \cdot \sum_{e\in E_b} f_e^\alpha$ where $E_b = \{(b,t) : f_{(b,t)} > \beta \cdot d\}$.

Recall that the total cost due to edges $E_a \cup E_b$ is $d \cdot \sum_{e\in E_a} f_e^\alpha + d \cdot \sum_{e\in E_b} f_e^\alpha \le \rho m d^\alpha$. This implies:

$$\sum_{e\in E_a} f_e + \sum_{e\in E_b} f_e \quad\le\quad \frac{\sum_{e\in E_a} f_e^\alpha + \sum_{e\in E_b} f_e^\alpha}{(\beta d)^{\alpha-1}} \quad\le\quad \frac{\rho\, m\, d^{\alpha-1}}{4\rho\, d^{\alpha-1}} \quad=\quad \frac{m}{4}.$$

Let $V_a = \{a \in A : |N_a| > 4\rho\beta,\ f_{(s,a)} \le \beta d\}$ and $V_b = \{b \in B : |N_b| > 4\rho\beta,\ f_{(b,t)} \le \beta d\}$. The total cost of edges $\{(a, a(u)) : a \in A, u \in L\} \cup \{(b(u), b) : b \in B, u \in L\}$ is at least $\sum_{a\in A} |N_a| \cdot d^{\alpha+1} + \sum_{b\in B} |N_b| \cdot d^{\alpha+1} \ge (|V_a| + |V_b|) \cdot 4\rho\beta\, d^{\alpha+1}$. Since the total routing cost is at most $\rho m d^\alpha$, we have $|V_a| + |V_b| \le \frac{\rho m d^\alpha}{4\rho\beta\, d^{\alpha+1}} = \frac{n}{4\beta}$. Thus $\sum_{w\in V_a} f_{(s,w)} + \sum_{w\in V_b} f_{(w,t)} \le (|V_a| + |V_b|) \cdot \beta d \le \frac{nd}{4} = \frac{m}{4}$.

Note that the flow lost in Step 1 above is at most $\sum_{e\in E_a} f_e + \sum_{w\in V_a} f_{(s,w)}$, and that in Step 2 is at most $\sum_{e\in E_b} f_e + + \sum_{w\in V_b} f_{(w,t)}$. So the total loss in flow is at most $\frac{m}{2}$. Finally consider Step 3. Let $E'' = \{e \in E' : f_e > \beta\}$. As in the calculation for E_a and E_b, using the fact that $\sum_{e\in E''} f_e^\alpha \le \rho\, m$ (since cost of edge $e \in E'$ is $d^\alpha \cdot f_e^\alpha$), we have $\sum_{e\in E''} f_e \le \frac{\rho\, m}{(\beta d)^{\alpha-1}} = \frac{m}{4}$. □

The flow after the above pruning has magnitude at least $\frac{m}{4}$ and edges in E' carry at most β flow each. If we choose one label for each vertex $c \in A \cup B$ randomly from N_c, the expected number of consistent edges in F is at least $\frac{m}{4\beta} \cdot \frac{1}{(4\beta\rho)^2} = \frac{m}{64\rho^2\beta^3} = \frac{|F|}{64\rho^{2+\frac{3}{\alpha-1}}}$.

Finally, the hardness of label cover implies that the s-t routing problem with any exponent $\alpha = 1 + \epsilon$ (for constant $\epsilon > 0$) is hard to approximate better than ratio $2^{\log^{1-\delta} n}$ for any $\delta > 0$. □

5.2 APX-Hardness of Undirected $s - t$ Routing with Heterogeneous Functions

We now consider the same case as the previous section, but in undirected graphs. Undirected graphs tend to be easier to route in than directed graphs, but we are still able to prove an inapproximability result.

Theorem 5. *The $s - t$ routing problem on undirected graphs with heterogeneous power functions and constant exponent $\alpha > 1$ is APX-Hard.*

Proof. The proof is a reduction from the problem of $\mathsf{3SAT}(2d)$, i.e., 3SAT where each variable appears in exactly $2d$ clauses (d as the positive form, and d as the negative form), for which the following hardness is known [14].

Theorem 6. *There exist constants d and ϵ for which it is NP-hard to distinguish between $\mathsf{3SAT}(2d)$ instances which are fully satisfiable and those which are at most $(1-\epsilon)$ satisfiable.*

From $\mathsf{3SAT}(2d)$ to Independent Set. We first reduce the problem of $\mathsf{3SAT}(2d)$, to that of finding large independent sets in bounded degree graphs. Indeed, given an instance of $\mathsf{3SAT}(2d)$ with $m = 2dn/3$ clauses on n variables, we construct the following graph: For each clause, we create a triangle with nodes corresponding to the literals. These triangles are disjoint across clauses. To tie up the instance, we place edges between x and $\overline{x}$ for all the occurrences of literals x and $\overline{x}$. Notice that there are $N := 3m$ variables, and $M := 3m + d^2n = N(1 + d/2)$ edges in this graph. Furthermore, each node has degree $d + 2$ (two edges in the triangle, and d edges to the opposite literal).

We now relate independent sets on this graph with satisfying assignments in the $\mathsf{3SAT}(2d)$ instance. In the yes case, suppose there is a fully satisfying assignment for the SAT instance. Then, we can pick one satisfying literal from each clause and it forms an independent set (there are no triangle edges picked, and because the assignment is consistent, there are no literal edges as well). The size of this independent set is $N/3 = m$ nodes.

In the opposite direction, suppose we have an independent set of size $\frac{N}{3}(1 - \epsilon) = m(1-\epsilon)$ nodes in the graph. Then, clearly, it has to pick at most one node from each of the clause triangles. Furthermore, these nodes must correspond to consistent literals (else we would include a literal edge). Therefore, the independent set naturally recovers an assignment which satisfies at least $m(1-\epsilon)$ clauses. We can therefore conclude this step with the following theorem:

Theorem 7. *There are constants d and ϵ for which it is NP-hard to distinguish between $d+2$-regular graphs on N nodes which have an independent set of size at least $N/3$ and those where all independent sets are at most $N/3(1-\epsilon)$.*

From Independent Set to Power-Routing. We now reduce from independent set instances to the routing instances. Indeed, given a $d+2$-regular graph on N nodes $G=(V,E)$, we create the following routing instance $H=(W,F)$.

Routing Instance H. For each edge $e \in E$, there is an edge-vertex $w_e \in V'$, and for each node $v \in V$, we have a node-vertex $w_v \in V'$. We connect each edge-vertex to the corresponding node-vertices (according to G). That is, if $e=(u,v)$ in G, then we connect w_e with w_u and w_v. These edges have a buy cost of 1 and no load cost. These are called the *intermediate edges* in our graph H. Likewise, we connect each node-vertex w_v vertex to a sink t, with edges of buy cost $B := \frac{1}{2(d+1)}$. Finally, we connect each edge-vertex w_e to a source s with cost l^α, where l is the load. For the demand, we require s to route M units of flow to t. In the remainder of the proof, we will use C to denote $(2N/3)$.

Lemma 6. *If there is a vertex cover of size $C := (2N/3)$ in G, there is a routing solution in H of total cost at most $BC+2M$.*

This is easy to see, as the source can send a unit flow along to each edge-vertex, which will then send the flow to the node which covers it (in the vertex cover in G), and finally this flow gets routed to the sink. For the soundness direction, we show the following lemma in the full version of the paper.

Lemma 7. *For large enough constant α, if there is routing in H of cost at most $(1+\epsilon')(2M+BC)$, then we can recover an "almost vertex cover" of size $C(1+\epsilon')+8\epsilon' M(d+1)$ in G. Here, an "almost vertex cover" is a collection of nodes incident on at least a $(1-10\epsilon'(d+2)^2)$ fraction of the edges.*

We now complete our proof. In particular, we will be interested in the case when $C := 2N/3$ (recall that it was hard to distinguish between independent sets of size $N/3$ and those of size $(1-\epsilon)N/3$ in G). Indeed, suppose there is an independent set of size $N/3$. Then there is a vertex cover of size $2N/3$, and by Lemma 6, there is a routing in H of cost at most $BC+2M$.

Now, in the other direction, suppose there is a routing of cost at most $(1+\epsilon')(BC+2M)$. Then by the above Lemma 7, we can recover a set of $C(1+\epsilon')+8\epsilon' M(d+1)$ nodes which are incident on $(1-10\epsilon'(d+2)^2)$ fraction of the edges. But we can extend this to a complete vertex cover by adding at most $10\epsilon'(d+2)^2 M$ nodes by picking one node from each of the uncovered edges. This implies there is an independent set of size at least $N/3 - O(\epsilon' d^2)N = N/3(1-\epsilon)$ nodes, for small enough constant ϵ' (recall that d is a constant determined in Thm 7). But by Theorem 7, this is impossible if $P \neq NP$, thus proving Theorem 5. □

6 Open Problems

Perhaps the most natural open question is whether there exists a poly-log competitive online algorithm for the case of multiple sources and multiple sinks.

Is the following simple algorithm good? Consider request (s_i, t_i). With probability $1/2^k$ "pretend" that the demand D is 2^k, and open all edges used in the cheapest way to route demand D from s_i to t_i assuming previous routes. Then route one unit of flow between s_i and t_i in the cheapest possible way along open edges.

References

1. Proceedings of the vision and roadmap workshop on routing telecom and data centers toward efficient energy use (October 2008)
2. Andrews, M., Antonakopoulos, S., Zhang, L.: Minimum-cost network design with (dis)economies of scale. In: FOCS, pp. 585–592 (2010)
3. Andrews, M., Fernández, A., Zhang, L., Zhao, W.: Routing for energy minimization in the speed scaling model. In: INFOCOM, pp. 2435–2443 (2010)
4. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O.: On-line routing of virtual circuits with applications to load balancing and machine scheduling. J. ACM 44(3), 486–504 (1997)
5. Benoit, A., Melhem, R., Renaud-Goud, P., Robert, Y.: Power-aware manhattan routing on chip multiprocessors. In: IEEE International Parallel and Distributed Processing Symposium (IPDPS) (May 2012)
6. Chakrabarty, D., Chekuri, C., Khanna, S., Korula, N.: Approximability of Capacitated Network Design. In: Günlük, O., Woeginger, G.J. (eds.) IPCO 2011. LNCS, vol. 6655, pp. 78–91. Springer, Heidelberg (2011)
7. Gafni, E.M., Bertsekas, D.P.: Path assignment for virtual circuit routing. In: Proceedings of the Symposium on Communications Architectures & Protocols, SIGCOMM 1983, pp. 21–25. ACM, New York (1983)
8. Gupta, A., Krishnaswamy, R., Pruhs, K.: Online primal-dual for non-linear optimization with applications to speed scaling. CoRR, abs/1109.5931 (2011)
9. Imase, M., Waxman, B.M.: Dynamic Steiner tree problem. SIAM J. Discrete Math. 4(3), 369–384 (1991)
10. Johnson, W.B., Schechtman, G., Zinn, J.: Best constants in moment inequalities for linear combinations of independent and exchangeable random variables. Ann. Probab. (1), 234–253 (1985)
11. Kim, J., Horowitz, M.A.: Adaptive supply serial links with sub-1-v operation and per-pin clock recovery. IEEE Journal of Solid-State Circuits 37(11), 1403–1413 (2002)
12. Raz, R.: A parallel repetition theorem. SIAM J. Comput. 27(3), 763–803 (1998)
13. Rosenthal, H.P.: On the subspaces of L^p $(p > 2)$ spanned by sequences of independent random variables. Israel J. Math. 8, 273–303 (1970)
14. Trevisan, L.: Non-approximability results for optimization problems on bounded degree instances. In: ACM Symposium on Theory of Computing, pp. 453–461 (2001)