Approximating k-Forest with Resource Augmentation: A Primal-Dual Approach

Eric Angel[1], Nguyen Kim Thang[1], and Shikha Singh[2]

[1] IBISC, University d'Evry Val d'Essonne, France.
{angel, thang}@ibisc.univ-evry.fr.
[2] Stony Brook University, Stony Brook, NY, USA.
shiksingh@cs.stonybrook.edu

Abstract. In this paper, we study the k-forest problem in the model of resource augmentation. In the k-forest problem, given an edge-weighted graph $G(V, E)$, a parameter k, and a set of m demand pairs $\subseteq V \times V$, the objective is to construct a minimum-cost subgraph that connects at least k demands. The problem is hard to approximate—the best-known approximation ratio is $O(\min\{\sqrt{n}, \sqrt{k}\})$. Furthermore, k-forest is as hard to approximate as the notoriously-hard densest k-subgraph problem.
While the k-forest problem is hard to approximate in the worst-case, we show that with the use of resource augmentation, we can efficiently approximate it up to a constant factor.
First, we restate the problem in terms of the number of demands that are *not* connected. In particular, the objective of the k-forest problem can be viewed as to remove at most $m - k$ demands and find a minimum-cost subgraph that connects the remaining demands. We use this perspective of the problem to explain the performance of our algorithm (in terms of the augmentation) in a more intuitive way.
Specifically, we present a polynomial-time algorithm for the k-forest problem that, for every $\varepsilon > 0$, removes at most $m - k$ demands and has cost no more than $O(1/\varepsilon^2)$ times the cost of an optimal algorithm that removes at most $(1 - \varepsilon)(m - k)$ demands.

1 Introduction

In the worst-case paradigm, algorithms for NP-hard problems are typically characterized by their *approximation ratio*, defined as the ratio between the worst-case cost of the algorithm and the cost of an all-powerful optimal algorithm. Many computationally-hard problems admit efficient worst-case approximations [30, 34, 49, 47]. However, there are several fundamental problems, such as k-densest subgraph [18, 4], set cover [38, 16], graph coloring [6, 48, 7], etc., for which no algorithm with a *reasonable* approximation guarantee is known.

Many problems that are hard in the worst-case paradigm admit simple and fast heuristics in practice. Illustrative examples include clustering problems (e.g.

This research was supported by the ANR project OATA n°ANR-15-CE40-0015-01 and the Chateaubriand Fellowship of the Office for Science & Technology of the Embassy of France in the United States.

k-median, k-means and correlation clustering) and SAT problems—simple algorithms and solvers for these NP-hard problems routinely find meaningful clusters [13] and satisfiable solutions [40] on practical instances respectively. A major direction in algorithmic research is to explain the gap between the observed practical performance and the provable worst-case guarantee of these algorithms. Previous work has looked at various approaches to analyze algorithms that rules out pathological worst-cases [36, 8, 50, 15]. One such widely-used approach, especially in the areas of online scheduling and matching [12, 42, 31, 32], is the model of *resource augmentation*.

In the resource-augmentation model, an algorithm is given some additional power and its performance is compared against that of an optimal algorithm without the additional power. Resource augmentation has been studied in various guises such as speed augmentation and machine augmentation (see Section 1.2 for details). Recently, Lucarelli et al. [37] unified the different notions of resource augmentation under a *generalized resource-augmentation model* that is based on LP duality. Roughly speaking, in the generalized resource-augmentation model, the performance of an algorithm is measured by the ratio between its worst-case objective value over the set of feasible solutions $\mathcal{P}$ and the optimal value which is constrained over a set $\mathcal{Q}$ that is a *strict* subset of $\mathcal{P}$. In other words, in the unified model, the algorithm is allowed to be optimized over relaxed constraints while the adversary (optimum) has tighter constraints.

Duality-based techniques have proved to be powerful tools in the area of online scheduling with resource augmentation. Since the seminal work of Anand et al. [1], many competitive algorithms have been designed for online scheduling problems [14, 25, 27, 26, 45, 2, 37]. Interestingly, the principle ideas behind the duality-based approach in the resource-augmentation setting are general and can be applied to other (non-scheduling, offline) optimization problems as well.

In this paper, we initiate the use of duality to analyze approximation algorithms with resource augmentation in the context of general optimization problems. We exemplify this approach by focusing on a problem that has no reasonable approximation in the worst-case paradigm—the *k-forest problem* [24].

The k-Forest Problem. In the k-forest problem, given an edge-weighted graph $G(V, E)$, a parameter k and a set of m demand pairs $\subseteq V \times V$, we need to find a minimum-cost subgraph that connects at least k demand pairs.

The k-forest problem is a generalization of the classic k-MST (minimum spanning tree) and the k-Steiner tree (with a common source) problems, both of which admit constant factor approximations. In particular, k-MST and k-Steiner tree can be approximated up to a factors of 2 and 4 respectively [11, 20]. On the other hand, the k-forest problem has resisted similar attempts—the best-known approximation guarantee is $O(\min\{\sqrt{n}, \sqrt{k}\})$ [22].

Hajiaghayi and Jain [24] show that the k-forest problem is roughly as hard as the celebrated *densest k-subgraph problem*. Given a graph G and a parameter k, the densest k-subgraph problem is to find a set of k vertices which induce the maximum number of edges. The densest k-subgraph problem has been studied extensively in the literature [18, 33, 4, 3, 44, 17, 5] and is regarded to be a hard

problem. Hajiaghayi and Jain [24] show that if there is a polynomial time r-approximation for the k-forest problem then there exists a polynomial time $2r^2$-approximation algorithm for the densest k-subgraph problem. The best known approximation guarantee for the densest k-subgraph problem is $O(n^{1/4+\varepsilon})$ [4]. Hajiaghayi and Jain [24] point out that an approximation ratio better than $O(n^{1/8})$ for k-forest (which implies an approximation ratio better than $O(n^{1/4})$ for densest k-subgraph) would require significantly new insights and techniques.

1.1 Our Approach and Contributions

We give the first polynomial-time constant-factor algorithm for the k-forest problem in the resource-augmentation model.

Our algorithm is based on the primal-dual algorithm by Hajiaghayi and Jain [24] for a closely-related problem, the *prize collecting generalized Steiner tree* (PCGST) problem. The k-forest problem is a Lagrangian relaxation of the PCGST problem [24]. Hajiaghayi and Jain [24] give a 3-approximation primal-dual algorithm for the PCGST problem. However, their algorithm is not *Lagrangian-multiplier preserving* [49], which makes it difficult to derive a constant-factor approximation for the k-forest problem. In this paper, we overcome the challenge posed by the non-Lagrangian-multiplier-preserving nature of the primal-dual algorithm by Hajiaghayi and Jain [24], to obtain a constant-factor approximation for the k-forest problem, by using resource augmentation.

The primal-dual approach is particularly well-suited to analyze algorithms with resource augmentation. In particular, the resource augmentation setting can be viewed as a game between an algorithm and the optimal (or the adversary) where the adversary is subject to tighter constraints. To apply this notion to the k-forest problem, we need a constraint to play this game between the algorithm and the adversary. A natural approach is to choose the number of demands connected as the comparative constraint. That is, the algorithm chooses to connect at least k "cheap" demands out of the total m demands while the adversary's requirement is higher—to connect slightly more than k demands. An alternate approach is to constrain the number of demands that each algorithm is allowed to ignore or remove, that is, the algorithm can remove up to $m-k$ "costly" demands while the adversary can remove slightly fewer demands. Note that with respect to exact and approximate solutions (without any resource augmentation), both approaches are equivalent.

In this paper, we use the framework of PCGST [24] and obtain our result by choosing the number of demands *that can be removed* as the constraint to be augmented. In particular, our algorithm for the k-forest problem can remove up to $m-k$ demands whereas the adversary can only remove up to $\lfloor(1-\varepsilon)(m-k)\rfloor$ demands. This tighter cardinality constraint allows the dual to "raise" an additional amount (depending on ε) to "pay" for the primal cost. We exploit this property to bound the cost of the algorithm's output and that of a dual feasible solution to derive the approximation ratio. We show the following.

Theorem 1. *There exists a polynomial-time algorithm for the k-forest problem that, for any $\varepsilon > 0$, removes at most $(m-k)$ connection demands and outputs a subgraph with cost at most $O(1/\varepsilon^2)$ times the cost of the subgraph output by the optimal algorithm that removes at most $\lfloor (1-\varepsilon)(m-k) \rfloor$ demands.*

Alternate LP Rounding Approach. In this paper, we use the primal-dual algorithm for the PCGST problem [24]. There also exists a LP-rounding based algorithm for the PCGST problem. In Section A, we show that a similar rounding scheme gives a constant apporximation for the k-forest problem as well. However, the rounding approach involves solving an LP of exponential size, which while being polynomial-time is still a signficant overhead on the running time. In contrast, our primal-dual algorithm is (a) light-weight and faster than the rounding scheme for dense graphs, making it practically appealing, and (b) gives a general framework which may prove useful for problems which do not admit rounding based solutions. We compare the two approaches in Section A.

Bi-criteria Approximation vs. Resource Augmentation. Although the result can be seen as a bi-criteria approximation, its interpretation is more meaningful in the sense of resource augmentation. While in multi-criteria optimization one tries to balance the qualities of different criteria, in resource augmentation the purpose is to design effective algorithms by violating some constraints by a factor as small as possible. Hence, the fact that an algorithm can approximate a hard problem with a small perturbation on the constraints would be an evidence to explain the performance of the algorithm in practice.

Augmentation Parameter: Demands Removed vs. Demands Connected. While the approach of connecting at least k demands is equivalent to rejecting up to $m-k$ demands with respect to exact and approximate solutions (without resource augmentation), there is a notable distinction between them in the presence of augmentation. In particular, allowing the adversary to remove up to $(1-\varepsilon)(m-k)$ demands (compared to $m-k$ demands removed by the algorithm), means we require the adversary to connect at least $k+\varepsilon(m-k)$ demands (compared to the k demands connected by the algorithm).

In this paper, we provide augmentation in terms of $m-k$, the number of demands that can be removed, because it leads to a more intuitive understanding of our algorithm's performance. In particular, our algorithm is *scalable* in terms of the parameter $m-k$, that is, it is a constant-factor approximation (depending on ε) with a factor $(1+\varepsilon)$ augmentation. On the other hand, in terms of the parameter k, our algorithm is a constant-factor approximation (depending on ε) with a factor $\left(1+\frac{m-k}{k}\cdot\varepsilon\right)$ augmentation, which is arguably not as insightful. We leave the question of obtaining a constant-factor approximation with a better augmentation in terms of k as an interesting open problem.

1.2 Additional Related Work

k-Forest and Variants. The k-forest problem generalizes both k-MST and k-Steiner tree. Chudak et al. [11] discuss the 2-approximation for k-MST [20] and give a 4-approximation for k-Steiner tree. Segev et al. [43] gave a

$O(\min\{n^{2/3}, \sqrt{m}\} \log n)$-approximation algorithm for the k-forest problem, which was improved by Gupta et al. [22] to a $O(\min \sqrt{n}, \sqrt{k})$-approximation. Gupta et al [22] also reduce a well-studied vehicle-routing problem in operations research, the *Dial-a-Ride* problem [9, 23, 19] to the k-forest problem. In particular, they show that an α-approximation for k-forest implies an $O(\alpha \log^2 n)$-approximation algorithm for the Dial-a-Ride problem.

Lagrangian Multiplier Preserving (LMP). This property [49] is desired when one designs algorithms in the prize-collecting settings. It is standard to transform a LMP algorithm to the one dealing with cardinality constraints. The illustrative examples consist of the algorithms for the k-median problem [29], k-MST problem [20], k-Steiner tree [11], partial covering problems [35]. Recall that the `HJ` algorithm is not LMP and that represents a difficulty to design algorithm for the k-forest problem.

Resource Augmentation and Duality. Kalyanasundaram and Pruhs [31] initiated the study of resource augmentation with the notion of *speed augmentation*, where an online scheduling algorithm is compared against an adversary with slower processing speed. Phillips et al. [42] proposed the *machine augmentation* model in which the algorithm has more machines than the adversary. Choudhury et al. [10] introduced the *rejection model* where an online scheduling algorithm is allowed to discard a small fraction of jobs. Many natural scheduling algorithms can be analyzed using these models and these analysis have provided theoretical evidence behind the practical performance of several scheduling heuristics. Recently, Lucarelli et al. [37] unified the different notions under a *generalized resource-augmentation model* using LP duality. To the best of our knowledge, such duality-based techniques have not been used in the context of approximation algorithms with resource augmentation.

2 Primal-Dual Algorithm for *k*-Forest

In this section, we present an efficient primal-dual algorithm for the k-forest problem in the resource-augmentation model.

In the k-forest problem, given an undirected graph $G(V, E)$ with a nonnegative cost c_e on each edge $e \in E$, a parameter k, and m connection demands $\mathcal{J} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\} \subseteq V \times V$, the objective is to construct a minimum-cost subgraph of G which connects at least k demands. To overcome the non-Lagrangian-multiplier-preserving barrier [24] and to take advantage of resource augmentation, we restate the problem as follows—given an undirected graph $G(V, E)$ with a nonnegative cost c_e on each edge $e \in E$, a parameter k, and m connection demands $\mathcal{J} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\} \subseteq V \times V$, the objective is remove up to $(m - k)$ demands and construct a minimum-cost subgraph of G that connects the remaining demands.

We use the algorithm by Hajiaghayi and Jain [24] for the prize-collecting generalized Steiner tree (PCGST) problem and refer to it by the shorthand `HJ`. In the prize-collecting generalized Steiner tree (PCGST) problem, given an undirected graph $G(V, E)$, with a nonnegative cost c_e on each edge $e \in E$,

m connection demands $\mathcal{J} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\}$ and a nonnegative penalty cost π_i for every demand $i \in \mathcal{J}$, the goal is to minimize the cost of buying a set of edges and paying a penalty for the demands that are not connected by the chosen edges. Without loss of generality, we can assume that $\mathcal{J} = V \times V$, as the penalty for demands that need not be connected can be set to zero.

Next, we restate the LP for the PCGST problem in terms of the k-forest problem and reproduce the relevant lemmas [24].

2.1 Hajiaghayi and Jain's LP for k-forest

Fix a constant $0 < \varepsilon < 1$. Set $\tilde{\varepsilon} = \varepsilon/2$ and set $r = (1 - \tilde{\varepsilon})(m - k)$. Let x_e be a variable such that $x_e = 1$ if edge $e \in E$ is included in the subgraph solution. Similarly, let z_i be a variable such that $z_i = 1$ if s_i, t_i are not connected in the subgraph solution. We restate the integer program for the PCGST problem [24] in terms of the k-forest problem in the resource augmentation model as $(\mathcal{P}_{\tilde{\varepsilon}})$.

$$\min \sum_{e \in E} c_e x_e \qquad (\mathcal{P}_{\tilde{\varepsilon}})$$

$$(y_{i,S}) \qquad \sum_{e \in \delta(S)} x_e + z_i \geq 1 \qquad \forall i, \forall S \subset V : S \odot i$$

$$(\lambda) \qquad \sum_{i,j \in V} z_i \leq (1 - \tilde{\varepsilon}) r$$

$$x_e, z_i \in \{0, 1\} \qquad \forall e \in E, \forall i$$

For a set $S \subset V$, the notation $S \odot i$ stands for $|\{s_i, t_i\} \cap S| = 1$. For a given non-empty set $S \subset V$, $\delta(S)$ denotes the set of edges defined by the *cut* S, that is, $\delta(S)$ is the set of all edges with exactly one endpoint in S. Thus, the first constraint says that for every cut $S \odot i$, there is at least one edge $e \in \delta(S)$ such that either edge e is included in the solution or demand i is removed. The second constraint says that the total number of demands removed is no more than $(1 - \tilde{\varepsilon})r$. Note that the optimal value of $(\mathcal{P}_{\tilde{\varepsilon}})$ is a lower bound on the optimal solution that removes at most $(1 - \varepsilon)(m - k)$ demands. This is because we have slightly relaxed the upper bound of the number of demands removed to be $(1 - \tilde{\varepsilon})r = (1 - \tilde{\varepsilon})^2(m - k) \geq (1 - \varepsilon)(m - k)$.

The dual $(\mathcal{D}_{\tilde{\varepsilon}})$ of the relaxation of $(\mathcal{P}_{\tilde{\varepsilon}})$ follows.

$$\max \sum_{S \subset V, S \odot i} y_{i,S} - (1 - \tilde{\varepsilon}) r \lambda \qquad (\mathcal{D}_{\tilde{\varepsilon}})$$

$$\sum_{S : e \in \delta(S), S \odot i} y_{i,S} \leq c_e \qquad \forall e \in E$$

$$\sum_{S : S \odot i} y_{i,S} \leq \lambda \qquad \forall i$$

$$y_{i,S} \geq 0 \qquad \forall S \subset V : S \odot i$$

Hajiaghayi and Jain [24] formulate a new dual $(\mathcal{D}^{\mathtt{HJ}}_{\tilde{\varepsilon}})$ equivalent to $(\mathcal{D}_{\tilde{\varepsilon}})$ based on Farkas lemma. This new dual resolves the challenges posed by raising different dual variables associated with the same set of vertices of the graph in $(\mathcal{D}_{\tilde{\varepsilon}})$. We refer the readers to the original paper [24] for a detailed discussion on the transformation and proofs.

Note that $\mathcal{S}$ is a *family* of subsets of V if $\mathcal{S} = \{S_1, S_2, \ldots, S_\ell\}$ where $S_j \subset V$ for $1 \leq j \leq \ell$. For a family $\mathcal{S}$, if there exists $S \in \mathcal{S}$ such that $S \odot i$, we denote it by $\mathcal{S} \odot i$. The new dual $(\mathcal{D}^{\mathtt{HJ}}_{\tilde{\varepsilon}})$ is stated below.

$$\begin{aligned}
\max \sum_{S \subset V} y_S - (1-\tilde{\varepsilon}) r\lambda \qquad & (\mathcal{D}^{\mathtt{HJ}}_{\tilde{\varepsilon}}) \\
\sum_{S: e \in \delta(S)} y_S \leq c_e \qquad & \forall e \in E \\
\sum_{S \in \mathcal{S}} y_S \leq \sum_{i, \mathcal{S} \odot i} \lambda \qquad & \forall \text{ family } \mathcal{S} \\
y_S \geq 0 \qquad & \forall S \subset V
\end{aligned}$$

We use the HJ algorithm (along with the construction of dual variables) for the PCGST problem. We set the penalty of every request to a fixed constant λ. We reproduce the relevant lemmas in terms of k-forest. See [24] for proofs.

For $S \subset V$, let $y_S(\lambda)$'s be the dual variables constructed in HJ algorithm with penalty cost λ. Let $y(\lambda)$ be the vector consisting of all $y_S(\lambda)$'s.

Lemma 1 ([24]). *Let $r(\lambda)$ be the number of demands removed with the penalty cost λ by the HJ algorithm. Then, $r(\lambda) \cdot \lambda \leq \sum_S y_S(\lambda)$.*

Lemma 2 ([24]). *Let F be the set of edges in the subgraph solution output by the HJ algorithm. Then $\sum_{e \in F} c_e \leq 2 \sum_S y_S(\lambda)$.*

2.2 Algorithm for k-Forest

Let $\mathtt{HJ}(\lambda)$ denote a call to the primal-dual algorithm of Hajiaghayi and Jain [24] for the PCGST problem with a penalty cost λ for every request. For a given value λ, let $r(\lambda)$ be the number of demands removed by the algorithm $\mathtt{HJ}(\lambda)$. Similar to the classic k-median algorithm [29], we do a binary search on the value of λ, and call the HJ as a subroutine each time. We describe our algorithm for k-forest next and refer to it as algorithm $\mathcal{A}$.

1. Let $c_{\min} = \min\{c_e : e \in E\}$. Initially set $\lambda^1 \leftarrow 0$ and $\lambda^2 \leftarrow \sum_{e \in E} c_e$.
2. While $(\lambda^2 - \lambda^1) > c_{\min}/m^2$, do the following:
   (a) Set $\lambda = (\lambda^1 + \lambda^2)/2$.
   (b) Call $\mathtt{HJ}(\lambda)$ and get $r(\lambda)$ (the number of demands removed).
      i. If $r(\lambda) = r$, then output the solution given by $\mathtt{HJ}(\lambda)$.
      ii. Otherwise, if $r(\lambda) < (1 - \varepsilon/2)r$ then update $\lambda^2 \leftarrow \lambda$;
      iii. Otherwise, if $r(\lambda) > r$ then update $\lambda^1 \leftarrow \lambda$.

3. Let α_1 and α_2 be such that $\alpha_1 r_1 + \alpha_2 r_2 = r$, $\alpha_1 + \alpha_2 = 1$ and $\alpha_1, \alpha_2 \geq 0$. Specifically,
$$\alpha_1 = \frac{r - r_2}{r_1 - r_2} \quad \text{and} \quad \alpha_2 = \frac{r_1 - r_0}{r_1 - r_2} \tag{1}$$
If $\alpha_2 \geq \tilde{\varepsilon}$, then return the solution $\texttt{HJ}(\lambda^2)$. Else, return the solution $\texttt{HJ}(\lambda^1)$.

Observe that the algorithm $\mathcal{A}$ always terminates: either it encounters a value of λ such that $r(\lambda) = r$ in Step 2(b)i or returns a solution depending on the final values of λ^1 and λ^2 in Step 3.

2.3 Analysis

Let $\textsc{Opt}_u$ be the cost of an optimal solution that removes at most u demands. Assume that $c_{\min} \leq \textsc{Opt}_{(1-\tilde{\varepsilon})r}$, because otherwise the optimal solution is to not select any edge $e \in E$. The algorithm outputs the solution either in Step 2(b)i or in Step 3. First, consider the case that the solution is output in Step 2(b)i.

Lemma 3. *Suppose that $\mathcal{A}$ outputs the solution given by $\texttt{HJ}(\lambda)$ in Step 2(b)i for some λ. Let F be the set of edges returned by $\texttt{HJ}(\lambda)$. Then,*
$$\sum_{e \in F} c_e \leq \frac{2}{\tilde{\varepsilon}} \cdot \textsc{Opt}_{(1-\tilde{\varepsilon})r}.$$

Proof. Since the solution is output in Step 2(b)i, the number of demands removed is $r(\lambda) = r$. By weak duality, the value of $\textsc{Opt}_{(1-\tilde{\varepsilon})r}$ is lower bounded by the objective cost of $(\mathcal{D}^{\texttt{HJ}}_{\tilde{\varepsilon}})$ with dual variables $y(\lambda)$. That is,
$$\textsc{Opt}_{(1-\tilde{\varepsilon})r} \geq \sum_{S \subset V} y_S - (1 - \tilde{\varepsilon}) r \lambda \geq \tilde{\varepsilon} \cdot \sum_{S \subset V} y_S \geq \frac{\tilde{\varepsilon}}{2} \cdot \sum_{e \in F} c_e$$
where the last two inequalities follow from Lemma 1 and 2 respectively. □

Next, consider the case that the solution is output in Step 3. Let F_1 and F_2 be the sets of edges returned by $\texttt{HJ}(\lambda^1)$ and $\texttt{HJ}(\lambda^2)$, respectively. Let r_1 and r_2 denote the number of demands removed by $\texttt{HJ}(\lambda^1)$ and $\texttt{HJ}(\lambda^2)$ respectively. Then, we have $\lambda^2 - \lambda^1 \leq c_{\min}/m^2$. As $c_{\min} \leq \textsc{Opt}_{(1-\tilde{\varepsilon})r}$, at the end of the while loop we have $\lambda^2 - \lambda^1 \leq c_{\min}/m^2 \leq \textsc{Opt}_{(1-\tilde{\varepsilon})r}/m^2$. Furthermore, $r_2 < r < r_1$.

Consider the dual vector (y^*, λ^*) defined as
$$(y^*, \lambda^*) = \alpha_1 (y(\lambda_1), \lambda_1) + \alpha_2 (y(\lambda_2), \lambda_2)$$
where the coefficients α_1 and α_2 are defined in Step 3 of algorithm $\mathcal{A}$. Then, (y^*, λ^*) forms a feasible solution to the dual $(\mathcal{D}^{\texttt{HJ}}_{\tilde{\varepsilon}})$ as it is a convex combination of two dual feasible solutions.

We bound the cost of algorithm $\mathcal{A}$ by bounding the cost of the dual $(\mathcal{D}^{\texttt{HJ}}_{\tilde{\varepsilon}})$.

Lemma 4. $\alpha_1 \sum_{e \in F_1} c_e + \alpha_2 \sum_{e \in F_2} c_e \leq \frac{4}{\tilde{\varepsilon}} \cdot \textsc{Opt}_{(1-\tilde{\varepsilon})r}$.

Proof. The cost of the dual ($\mathcal{D}^{\texttt{HJ}}_{\tilde{\varepsilon}}$) lower bounds the cost of an optimal algorithm that removes at most $(1-\tilde{\varepsilon})r$ demands. That is,

$$
\begin{aligned}
&\textsc{Opt}_{(1-\tilde{\varepsilon})r} \geq \left(\sum_S y_S(\lambda^*) - (1-\tilde{\varepsilon})r\lambda^*\right) \\
&= \alpha_1\left(\sum_S y_S(\lambda_1) - (1-\tilde{\varepsilon})r_1\lambda^*\right) + \alpha_2\left(\sum_S y_S(\lambda_2) - (1-\tilde{\varepsilon})r_2\lambda^*\right) \\
&= \alpha_1\left(\sum_S y_S(\lambda_1) - (1-\tilde{\varepsilon})r_1\lambda_1\right) - \alpha_1(1-\tilde{\varepsilon})r_1(\lambda^* - \lambda_1) \\
&\qquad + \alpha_2\left(\sum_S y_S(\lambda_2) - (1-\tilde{\varepsilon})r_2\lambda_2\right) + \alpha_2(1-\tilde{\varepsilon})r_2(\lambda_2 - \lambda^*) \\
&\geq \alpha_1\left(\sum_S y_S(\lambda_1) - (1-\tilde{\varepsilon})r_1\lambda_1\right) + \alpha_2\left(\sum_S y_S(\lambda_2) - (1-\tilde{\varepsilon})r_2\lambda_2\right) - m(\lambda^* - \lambda_1) \qquad (2) \\
&\geq \tilde{\varepsilon}\Bigg[\alpha_1\frac{1}{\tilde{\varepsilon}}\left(\sum_S y_S(\lambda_1) - (1-\tilde{\varepsilon})r_1\lambda_1\right) \\
&\qquad\qquad + \alpha_2\frac{1}{\tilde{\varepsilon}}\left(\sum_S y_S(\lambda_2) - (1-\tilde{\varepsilon})r_2\lambda_2\right)\Bigg] - \frac{\textsc{Opt}_{(1-\tilde{\varepsilon})r}}{m} \qquad (3) \\
&= \tilde{\varepsilon}\alpha_1\left[\left(\frac{1}{\tilde{\varepsilon}} - 1\right)\left(\sum_S y_S(\lambda_1) - r_1\lambda_1\right) + \sum_S y_S(\lambda_1)\right] \\
&\qquad\qquad + \tilde{\varepsilon}\alpha_2\left[\left(\frac{1}{\tilde{\varepsilon}} - 1\right)\left(\sum_S y_S(\lambda_2) - r_2\lambda_2\right) + \sum_S y_S(\lambda_2)\right] - \frac{\textsc{Opt}_{(1-\tilde{\varepsilon})r}}{m} \\
&\geq \tilde{\varepsilon}\left(\alpha_1\sum_S y_S(\lambda_1) + \alpha_2\sum_S y_S(\lambda_2)\right) - \frac{\textsc{Opt}_{(1-\tilde{\varepsilon})r}}{m} \qquad (4) \\
&\geq \frac{\tilde{\varepsilon}}{2}\left(\alpha_1\sum_{e\in F_1} c_e + \alpha_2\sum_{e\in F_2} c_e\right) - \frac{\textsc{Opt}_{(1-\tilde{\varepsilon})r}}{m} \qquad (5)
\end{aligned}
$$

Inequality (2) holds because $\lambda_1 \leq \lambda^* \leq \lambda_2$, $r_1 < m$, $0 \leq \alpha_1, \alpha_2 \leq 1$ and $0 < \tilde{\varepsilon} < 1$. Inequality (3) follows from the definition of the penalty costs, that is, $\lambda^* - \lambda_1 \leq \lambda_2 - \lambda_1 \leq \textsc{Opt}_{(1-\tilde{\varepsilon})r}/m^2$. Inequality (4) follows from Lemma 1 and the fact that $1/\tilde{\varepsilon} - 1 > 0$. Finally, Inequality (5) uses Lemma 2.

Rearranging the terms of Inequality (5) proves Lemma 4, that is,

$$\alpha_1\sum_{e\in F_1} c_e + \alpha_2\sum_{e\in F_2} c_e \leq \frac{2}{\tilde{\varepsilon}}\cdot\frac{m+1}{m}\cdot\textsc{Opt}_{(1-\tilde{\varepsilon})r} \leq \frac{4}{\tilde{\varepsilon}}\cdot\textsc{Opt}_{(1-\tilde{\varepsilon})r}.$$

□

We are now ready to prove the main theorem.

Proof of Theorem 1. We analyze algorithm $\mathcal{A}$. Lemma 3 is sufficient for the case that $\mathcal{A}$ outputs the solution in Step 2(b)i. Now suppose that $\mathcal{A}$ outputs the solution in Step 3.

Note that $(1-\tilde{\varepsilon})r \geq (1-\varepsilon)(m-k) \geq \lfloor (1-\varepsilon)(m-k) \rfloor$, therefore, we have,

$$\textsc{Opt}_{(1-\tilde{\varepsilon})r} \leq \textsc{Opt}_{\lfloor (1-\varepsilon)(m-k) \rfloor}.$$

We consider two cases based on the value of α_2.

Case 1: $\alpha_2 \geq \tilde{\varepsilon}$**.** $\mathcal{A}$ returns F_2 which is a feasible solution since the number of demands removed is $r_2 \leq r$. We bound the cost of solution F_2 using Lemma 4:

$$\begin{aligned}\sum_{e \in F_2} c_e &\leq \frac{1}{\tilde{\varepsilon}} \alpha_2 \sum_{e \in F_2} c_e \leq \frac{1}{\tilde{\varepsilon}} \left(\alpha_1 \sum_{e \in F_1} c_e + \alpha_2 \sum_{e \in F_2} c_e \right) \\ &\leq \frac{4}{\tilde{\varepsilon}^2} \cdot \textsc{Opt}_{(1-\tilde{\varepsilon})r} \leq \frac{4}{\tilde{\varepsilon}^2} \cdot \textsc{Opt}_{\lfloor (1-\varepsilon)(m-k) \rfloor}.\end{aligned}$$

Case 2: $\alpha_2 < \tilde{\varepsilon}$**.** $\mathcal{A}$ outputs F_1 as the solution. Since $\alpha_1 + \alpha_2 = 1$ by definition, we have $\alpha_1 > 1 - \tilde{\varepsilon}$. Using equation (1), we have:

$$r - r_2 \geq (1-\tilde{\varepsilon})(r_1 - r_2) \Rightarrow r - \tilde{\varepsilon} r_2 \geq (1-\tilde{\varepsilon}) r_1 \Rightarrow r_1 \leq \frac{1}{(1-\tilde{\varepsilon})} \cdot r = (m-k)$$

where the last equality uses $r = (1-\tilde{\varepsilon})(m-k)$. Thus, F_1 is a feasible solution.

We bound the cost of solution F_1, applying Lemma 4 again:

$$\begin{aligned}\sum_{e \in F_1} c_e &\leq \frac{1}{1-\tilde{\varepsilon}} \alpha_1 \sum_{e \in F_1} c_e \leq \frac{1}{1-\tilde{\varepsilon}} \left(\alpha_1 \sum_{e \in F_1} c_e + \alpha_2 \sum_{e \in F_2} c_e \right) \\ &\leq \frac{4}{\tilde{\varepsilon}^2} \cdot \textsc{Opt}_{(1-\tilde{\varepsilon})r} \leq \frac{4}{\tilde{\varepsilon}^2} \cdot \textsc{Opt}_{\lfloor (1-\varepsilon)(m-k) \rfloor}\end{aligned}$$

where the third inequality holds since $(1-\tilde{\varepsilon}) \geq 1/2 \geq \tilde{\varepsilon}$.

The two cases together prove the approximation and augmentation factors of $\mathcal{A}$ in Theorem 1 (recall that $\tilde{\varepsilon} = \varepsilon/2$). □

We now analyze the exact running of algorithm $\mathcal{A}$.

Lemma 5. *The running time of algorithm $\mathcal{A}$ is $O\big(n^6 \log(\frac{1}{\varepsilon} m) T\big)$, where T is the maximum number of bits required to represent the edge weights of the graph.*

Proof. In the `HJ` algorithm the main loop of the algorithm terminates in $O(n)$ steps and the most expensive part of the computation in each iteration involves linear number of maxflow computations where the graph is a bipartite graph with vertices corresponding to active components on one side and $V \times V$ on the other side. As the number of active components in the `HJ` algorithm is at most n, we get that the bipartite graph at any time step has at most n^2+n vertices and at most n^3 edges. Moreover, the maxflow computation in a (unweighted) bipartite graph is equivalent to computing maximum (cardinal) matching. The latter can be computed in time $O(\sqrt{|V|} \cdot |E|)$ [39] where V and E are the sets of vertices and edges in the graph. Thus, the overall running time of the `HJ` algorithm is $O(n^5)$. See [24] for details.

Algorithm $\mathcal{A}$ makes $O\big(\log(\frac{1}{\varepsilon} m^2 \frac{\sum_e c_e}{c_{\min}})\big)$ calls to the `HJ` algorithm. Thus, the running time of $\mathcal{A}$ is $O\big(n^5 \cdot \log(m/\varepsilon) \cdot T\big)$. □

3 Conclusion

The model of resource augmentation has been widely-used and has successfully provided theoretical evidence for several heuristics, especially in the case of online scheduling problems. Surprisingly, for offline algorithms, not many scalable approximation algorithms have been designed, despite the need of effective algorithms for hard problems.

In this paper, we initiate the study of hard (to approximate) problems in the resource-augmentation model. We show that the k-forest problem can be approximated up to a constant factor using augmentation. It is an interesting direction to design algorithms in the resource augmentation model for other hard problems which currently admit no meaningful approximation guarantees.

Acknowledgments

We thank Samuel McCauley for giving us his valuable feedback. We thank an anonymous reviewer for suggesting the rounding algorithm given in Section A.

References

1. Anand, S., Garg, N., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: Proc. 23rd Symposium on Discrete Algorithms. pp. 1228–1241 (2012)
2. Angelopoulos, S., Lucarelli, G., Thang, N.K.: Primal-dual and dual-fitting analysis of online scheduling algorithms for generalized flow time problems. In: Proc. 23rd European Symposium on Algorithms. pp. 35–46 (2015)
3. Asahiro, Y., Iwama, K., Tamaki, H., Tokuyama, T.: Greedily finding a dense subgraph. Journal of Algorithms 34(2), 203–221 (2000)
4. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In: Proc. 42nd Symposium on Theory of Computing. pp. 201–210 (2010)
5. Birnbaum, B., Goldman, K.J.: An improved analysis for a greedy remote-clique algorithm using factor-revealing lps. Algorithmica 55(1), 42–59 (2009)
6. Blum, A.: New approximation algorithms for graph coloring. Journal of the ACM 41(3), 470–516 (1994)
7. Blum, A., Karger, D.: An õ (n314)-coloring algorithm for 3-colorable graphs. Information processing letters 61(1), 49–53 (1997)
8. Borodin, A., Irani, S., Raghavan, P., Schieber, B.: Competitive paging with locality of reference. Journal of Computer and System Sciences 50(2), 244–258 (1995)
9. Charikar, M., Raghavachari, B.: The finite capacity dial-a-ride problem. In: Proc. 39th Symposium on Foundations of Computer Science. pp. 458–467 (1998)
10. Choudhury, A.R., Das, S., Garg, N., Kumar, A.: Rejecting jobs to minimize load and maximum flow-time. In: Proc. 26th Symposium on Discrete Algorithms. pp. 1114–1133 (2015)
11. Chudak, F.A., Roughgarden, T., Williamson, D.P.: Approximate k-msts and k-steiner trees via the primal-dual method and lagrangean relaxation. In: Proc. 8th Conference on Integer Programming and Combinatorial Optimization. pp. 60–70 (2001)

12. Chung, C., Pruhs, K., Uthaisombut, P.: The online transportation problem: On the exponential boost of one extra server. In: Proc. 8th Latin American Symposium on Theoretical Informatics. pp. 228–239 (2008)
13. Daniely, A., Linial, N., Saks, M.: Clustering is difficult only when it does not matter. arXiv preprint arXiv:1205.4891 (2012)
14. Devanur, N.R., Huang, Z.: Primal dual gives almost optimal energy efficient online algorithms. In: Proc. 25th Symposium on Discrete Algorithms (2014)
15. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. In: Proc. 36th International Colloquium on Automata, Languages, and Programming. pp. 427–438 (2009)
16. Feige, U.: A threshold of ln n for approximating set cover (preliminary version). In: Proc. 28th Symposium on Theory of Computing. pp. 314–318 (1996)
17. Feige, U., Langberg, M.: Approximation algorithms for maximization problems arising in graph partitioning. Journal of Algorithms 41(2), 174–211 (2001)
18. Feige, U., Peleg, D., Kortsarz, G.: The dense k-subgraph problem. Algorithmica 29(3), 410–421 (2001)
19. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: Proc. 17th Symposium on Foundations of Computer Science. pp. 216–227 (1976)
20. Garg, N.: A 3-approximation for the minimum tree spanning k vertices. In: Proc. 37th Symposium on Foundations of Computer Science. pp. 302–309 (1996)
21. Goemans, M.X., Williamson, D.P.: A general approximation technique for constrained forest problems. SIAM Journal on Computing 24(2), 296–317 (1995)
22. Gupta, A., Hajiaghayi, M., Nagarajan, V., Ravi, R.: Dial a ride from k-forest. ACM Transactions on Algorithm 6(2), 41 (2010)
23. Haimovich, M., Rinnooy Kan, A.: Bounds and heuristics for capacitated routing problems. Mathematics of operations Research 10(4), 527–542 (1985)
24. Hajiaghayi, M.T., Jain, K.: The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema. In: Proc. 17th Symposium on Discrete Algorithm. pp. 631–640 (2006)
25. Im, S., Kulkarni, J., Munagala, K.: Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In: Proc. 46th Symposium on Theory of Computing (2014)
26. Im, S., Kulkarni, J., Munagala, K.: Competitive flow time algorithms for polyhedral scheduling. In: Proc. 56th Symposium on Foundations of Computer Science. pp. 506–524 (2015)
27. Im, S., Kulkarni, J., Munagala, K., Pruhs, K.: Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In: Proc. 55th Symposium on Foundations of Computer Science (2014)
28. Jain, K.: A factor 2 approximation algorithm for the generalized steiner network problem. Combinatorica 21(1), 39–60 (2001)
29. Jain, K., Vazirani, V.V.: Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. Journal of the ACM 48(2), 274–296 (2001)
30. Johnson, D.S.: Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences 9(3), 256–278 (1974)
31. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. Journal of the ACM 47(4), 617–643 (2000)
32. Kalyanasundaram, B., Pruhs, K.R.: The online transportation problem. SIAM Journal on Discrete Mathematics 13(3), 370–383 (2000)

33. Khot, S.: Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. SIAM Journal on Computing 36(4), 1025–1071 (2006)
34. Klein, P.N., Young, N.E.: Approximation algorithms for NP-hard optimization problems. Chapman & Hall (2010)
35. Könemann, J., Parekh, O., Segev, D.: A unified approach to approximating partial covering problems. Algorithmica 59(4), 489–509 (2011)
36. Koutsoupias, E., Papadimitriou, C.H.: Beyond competitive analysis. SIAM Journal on Computing 30(1), 300–317 (2000)
37. Lucarelli, G., Thang, N.K., Srivastav, A., Trystram, D.: Online non-preemptive scheduling in a resource augmentation model based on duality. In: Proc. 24th European Symposium on Algorithms (2016)
38. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. Journal of the ACM 41(5), 960–981 (1994)
39. Micali, S., Vazirani, V.V.: An $O(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs. In: Proc. 21st Symposium on Foundations of Computer Science. pp. 17–27 (1980)
40. Ohrimenko, O., Stuckey, P.J., Codish, M.: Propagation via lazy clause generation. Constraints 14(3), 357–391 (2009)
41. Orlin, J.B.: Max flows in $O(nm)$ time, or better. In: Proc. 45th ACM Symposium on Theory of computing. pp. 765–774. ACM (2013)
42. Phillips, C.A., Stein, C., Torng, E., Wein, J.: Optimal time-critical scheduling via resource augmentation. Algorithmica 32(2), 163–200 (2002)
43. Segev, D., Segev, G.: Approximate k-steiner forests via the lagrangian relaxation technique with internal preprocessing. Algorithmica 56(4), 529–549 (2010)
44. Srivastav, A., Wolf, K.: Finding dense subgraphs with semidefinite programming. In: Workshop on Approximation Algorithms for Combinatorial Optimization. pp. 181–191 (1998)
45. Thang, N.K.: Lagrangian duality in online scheduling with resource augmentation and speed scaling. In: Proc. 21st European Symposium on Algorithms. pp. 755–766 (2013)
46. Vaidya, P.M.: A new algorithm for minimizing convex functions over convex sets. In: Foundations of Computer Science, 1989., 30th Annual Symposium on. pp. 338–343. IEEE (1989)
47. Vazirani, V.V.: Approximation Algorithms. Springer Science & Business Media (2013)
48. Wigderson, A.: Improving the performance guarantee for approximate graph coloring. Journal of the ACM 30(4), 729–735 (1983)
49. Williamson, D.P., Shmoys, D.B.: The design of approximation algorithms. Cambridge University Press (2011)
50. Young, N.E.: On-line paging against adversarially biased random inputs. Journal of Algorithms 37(1), 218–235 (2000)

A Alternate LP Rounding Based Algorithm

In this section, we describe a conceptually simple rounding algorithm for the k-forest problem. Fix a arbitrarily small constant $\varepsilon > 0$.

1. Solve the LP $(\mathcal{P}_0)$. Let (x^*, z^*) be a optimal fractional solution.

2. Remove all the demands i such that $z_i^* > 1 - \varepsilon$. Let L be the set of the remaining demands.
3. Apply the Goemans-Williamson primal-dual algorithm [21] on the set of remaining demands L and return the solution.

This algorithm is polynomial since there is a standard separation oracle based on the maximum flow to solve the the LP $(\mathcal{P}_0)$. Specifically, the separation oracle for the constraint $\sum_{e\in\delta(S)} x_e + z_i \geq 1$ can be done as follows. Given a solution (x, z), construct a network flow problem on the given graph G in which the capacity of each edge e is x_e. Then, for every i, verify if the maximum flow from s_i to t_i is at least $1 - z_i$. If not, then the minimum cut S separating s_i and t_i gives the violated constraint $\sum_{e\in\delta(S)} x_e + z_i < 1$. Otherwise, $\sum_{e\in\delta(S)} x_e + z_i \geq 1$ by the maxflow-mincut theorem. Hence, given a solution (x, z), one can find a violated constraint in polynomial time if it exists.

However, as it involves solving an LP of exponential size, in practice it is less performant than the primal-dual one presented in the main part of this paper.

Proposition 1. *The algorithm removes at most $(1+\varepsilon)(m-k)$ demands and has cost at most $O(1/\varepsilon)$ that of the optimal solution that removes $(m-k)$ demands.*

Proof. By the constraint $\sum_i z_i \leq r = (m-k)$ of $(\mathcal{P}_0)$, the number of variables z_i^*'s such that $z_i^* > 1 - \varepsilon$ is at most $(m-k)/(1-\varepsilon) \approx (1+\varepsilon)(m-k)$. So the number of removing demands is at most $(1+\varepsilon)(m-k)$. As $z_i^* \leq 1 - \varepsilon$ for all remaining demands $i \in L$, x^* is now a feasible solution of the following LP.

$$\begin{aligned} \min \sum_{e\in E} c_e x_e & \\ \sum_{e\in\delta(S)} x_e \geq \varepsilon & \qquad \forall i \in L, \forall S \subset V : S \odot i \\ x_e \geq 0 & \qquad \forall e \in E \end{aligned}$$

This is exactly the LP relaxation of the classic Steiner Forest problem by scaling up the constraints by factor $1/\varepsilon$. The Goemans-Williamson primal-dual algorithm [21] gives a 2 approximation for the latter problem. As x^* is a feasible solution of the LP above, the returned solution has cost at most $2/\varepsilon \cdot \sum_{e\in E} c_e x_e^*$. So the proposition follows. □

Running time of the rounding solution. The running time of the rounding scheme follows the analysis of Jain [28]. In particular, they show that the separation oracle for the problem can be implemented in time $O(nM(m, n))$ time, where $M(m, n)$ is the running time of maximum flow compuation in the the graph $G = (V, E)$ with n vertices and m edges. This can be plugged in into the running time of Vaidya's algorithm [46] to solve the LP relaxation. This gives us the running time of finding the optimal solution of the LP as $O(m^2 n(T + \log m)M(m, n) + m^2(T + \log m)P(m))$, where $P(m)$ is the time to

multiply two $m \times m$ matrices.

Rounding vs. Primal-Dual Approach for the k-Forest Problem. To compare the two approaches, we first compare their running times.

Using the Orlin maxflow algorithm [41], we have $M(m,n) = mn$. Hence, the complexity of the rounding algorithm is $O(m^3n^2T)$ whereas the running time of the primal-dual based Algorithm $\mathcal{A}$ is $O\big(n^6 \cdot \log(m/\varepsilon) \cdot T\big)$ using Lemma 5.

Thus, for sufficiently dense-graphs—in particular when $m > n^{4/3} \log n$—the primal-dual algorithm outperforms the rounding algorithm.

Second, we note that the rounding approach requires solving an exponetial-size LP, which in general is not practical. Light-weight algorithms such as greedy or primal-dual routinely outperform exponential-size rounding-based algorithms.

Finally, the primal-dual approach establishes a general technique which can prove useful in solving other non-Langrangian-multiplier preserving optimization problems that may not admit efficient rounding based solutions.