Parallel Approximation Algorithms for Facility-Location Problems

Guy E. Blelloch   Kanat Tangwongsan

Carnegie Mellon University
{guyb, ktangwon}@cs.cmu.edu

ABSTRACT

This paper presents the design and analysis of parallel approximation algorithms for facility-location problems, including NC and RNC algorithms for (metric) facility location, k-center, k-median, and k-means. These problems have received considerable attention during the past decades from the approximation algorithms community, which primarily concentrates on improving the approximation guarantees. In this paper, we ask: *Is it possible to parallelize some of the beautiful results from the sequential setting?*

Our starting point is a small, but diverse, subset of results in approximation algorithms for facility-location problems, with a primary goal of developing techniques for devising their efficient parallel counterparts. We focus on giving algorithms with low depth, near work efficiency (compared to the sequential versions), and low cache complexity.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms, Theory

Keywords

Parallel algorithms, approximation algorithms, facility location problems

1. INTRODUCTION

Facility location is an important and well-studied class of problems in approximation algorithms, with far-reaching implications in areas as diverse as machine learning, operations research, and networking: the popular k-means clustering and many network-design problems are all examples of problems in this class. Not only are these problems important because of their practical value, but they appeal to study

because of their special stature as "testbeds" for techniques in approximation algorithms. Recent research has focused primarily on improving the approximation guarantee, producing a series of beautiful results, some of which are highly efficient—often, with the sequential running time within constant or polylogarithmic factors of the input size.

Despite significant progress on these fronts, work on developing parallel approximation algorithms for these problems remains virtually non-existent. Although variants of these problems have been considered in the the distributed computing setting [29, 11, 32], to the best our of knowledge, almost no prior work has looked directly in the parallel setting where the total work and parallel time (depth) are the parameters of concern. The only prior work on these problems is due to Wang and Cheng, who gave a 2-approximation algorithm for k-center that runs in $O(n \log^2 n)$ depth and $O(n^3)$ work [37], a result which we improve in this paper.

Deriving parallel algorithms for facility location problems is a non-trivial task and will be a valuable step in understanding how common techniques in approximation algorithms can be parallelized efficiently. Previous work on facility location commonly relies on techniques such as linear-program (LP) rounding, local search, primal dual, and greedy. Unfortunately, LP rounding relies on solving a class of linear programs not known to be solvable efficiently in polylogarithmic time. Neither do known techniques allow for parallelizing local-search algorithms. Despite some success in parallelizing primal-dual and greedy algorithms for set-covering, vertex-covering, and related problems, these algorithm are obtained using problem-specific techniques, which are not readily applicable to other problems.

Summary of Results. In this paper, we design and analyze several algorithms for (metric) facility location, k-median, k-means and k-center problems, focusing on parallelizing a diverse set of techniques in approximation algorithms. We study the algorithms on the EREW PRAM and the Parallel Cache Oblivious model [3]. The latter model captures memory locality. We are primarily concerned with minimizing the work (or cache complexity) while achieving polylogarithmic depth in these models. We are less concerned with polylogarithmic factors in the depth since such measures are not robust across models. By work, we mean the total operation count. All algorithms we develop are in NC or RNC, so they have polylogarithmic depth.

We first present a parallel RNC algorithm mimicking the greedy algorithm of Jain et al. [18]. This is the most challenging algorithm to parallelize because the greedy algorithm is inherently sequential. We show the algorithm gives a

$(6 + \varepsilon)$-approximation and does $O(m \log_{1+\varepsilon}^2 m)$ work, which is within a logarithmic factor of the serial algorithm. Then, we present a simple RNC algorithm using the primal-dual approach of Jain and Vazirani [17] which leads to a $(3 + \varepsilon)$-approximation and for input of size m runs in $O(m \log_{1+\varepsilon} m)$ work, which is the same as the sequential work. The sequential algorithm is a 3-approximation. Following that, we present a local-search algorithm for k-median and k-means, with approximation factors of $5 + \varepsilon$ and $81 + \varepsilon$, matching the guarantees of the sequential algorithms. For constant k, the algorithm does $O(n^2 \log n)$ work, which is the same as the sequential counterpart. Furthermore, we present a 2-approximation algorithm for k-center with $O((n \log n)^2)$ work, based on the algorithm of Hochbaum and Shmoys [16]. Finally, we show a $O(m \log_{1+\varepsilon}^2(m))$-work randomized rounding algorithm, which yields a $(4 + \varepsilon)$-approximation, given an optimal linear-program solution as input. The last two algorithms run in work within a logarithmic factor of the serial algorithm counterparts.

Related Work. Facility-location problems have had a long history. Because of space consideration, we mention only some of the results here, focusing on those concerning metric instances. For the (uncapacitated) metric facility location, the first constant factor approximation was given by Shmoys et al. [34], using an LP-rounding technique, which has subsequently been improved [7, 13]. A different approach, based on local-search techniques, has been used to obtain a 3-approximation [22, 1, 14]. Combinatorial algorithms based on primal-dual and greedy approaches with constant approximation factors are also known [18, 17, 31]. Other approximation algorithms and hardness results have also been given by [36, 8, 4, 5, 27, 28, 22, 5, 13]. An open problem is to close the gap between the best known approximation factor of 1.5 [4] and the hardness result of 1.463 [13].

The first constant factor approximation for k-median problem was given by Charikar et al. [6], which was subsequently improved by [5] and [1] to the current best factor of $3 + \varepsilon$. For k-means, constant-factor approximations are known for this problem [17, 14]; a special case when the metric space is the Euclidean space has also been studied [20]. For k-center, tight bounds are known: there is a 2-approximation algorithm due to [12, 15], and this is tight unless $\mathsf{P} = \mathsf{NP}$.

The study of parallel approximation algorithms has been slow since the early 1990s. There are RNC and NC parallel approximation algorithms for set cover [2, 33], vertex cover [21, 23], special cases of linear programs (e.g., positive LPs and cover-packing LPs) [26, 35, 38], and k-center [37]. These algorithms are typically based on parallelizing their sequential counterparts, which usually contain an inherently sequential component (e.g., a greedy step which requires picking and processing the minimum-cost element before proceeding to the next). A common idea in these parallel algorithms is that instead of picking only the most cost-effective element, they make room for parallelism by allowing a small slack (e.g., a $(1 + \varepsilon)$ factor) in what can be selected. This idea often results in a slightly worse approximation factor than the sequential version. For instance, the parallel set-cover algorithm of Rajagopalan and Vazirani is a $(2(1 + \varepsilon) \ln n)$-approximation, compared to a $(\ln n)$-approximation produced by the standard greedy set cover. Likewise, the parallel vertex-cover algorithm of Khuller et al. is a $2/(1 - \varepsilon)$-approximation as opposed to the optimal 2-approximation given by various known sequential algorithms. Only recently has the approx-

imation factor for vertex cover been improved to 2 in the parallel case [23].

Several approximation algorithms have been proposed for distributed computing; see, e.g. [10], for a survey. For facility location, recent research has proposed a number of algorithms, both for the metric and non-metric cases [29, 11, 32]. The work of Pandit and Pemmaraju [32] is closely related our primal-dual algorithm; their algorithm is a 7-approximation in the CONGEST model for distributed computing. Both their algorithm and ours have a similar preprocessing step and rely on the $(1 + \varepsilon)$-slack idea although their algorithm uses a fixed $\varepsilon = 1$. The model and the efficiency metrics studied are different, however.

2. PRELIMINARIES AND NOTATION

Let F denote a set of *facilities* and C denote a set of *clients*. For convenience, let $n_c = |C|$, $n_f = |F|$, and $m = n_c \times n_f$. Each facility $i \in F$ has a cost of f_i, and each client $j \in C$ incurs a cost ("distance") $d(j, i)$ to use the facility i. We assume throughout that there is a metric space (X, d) with $F \cup C \subseteq X$ that underlies our problem instances. Thus, the distance d is symmetric and satisfies the triangle inequality. As a shorthand, denote the cost of the optimal solution by opt, the facility set of the optimal solution by F^*, and the facility set produced by our algorithm by F_A. Furthermore, we write $d(u, S)$ to mean the minimum distance from u to a member of S, i.e., $d(u, S) = \min\{d(u, w) : w \in S\}$.

Let G be a graph. We denote by $\deg_G(v)$ the degree of the node v in G and use $\Gamma_G(v)$ to denote the neighbor set of the node v. We drop the subscript (i.e., writing $\deg(v)$ and $\Gamma(v)$) when the context is clear. Let $V(G)$ and $E(G)$ denote respectively the set of nodes and the set of edges.

Parallel Models. All the parallel algorithms in this paper can be expressed in terms of a set of simple operations on vectors and dense matrices, making it easy to analyze costs on a variety of parallel models. In particular, the distances $d(\cdot, \cdot)$ can be represented as a dense $n \times n$ matrix, where $n = n_c + n_f$, and any data at clients or facilities can be represented as vectors. The only operations we need are parallel loops over the elements of the vector or matrix, transposing the matrix, sorting the rows of a matrix, and summation, prefix sums and distribution across the rows or columns of a matrix or vector. A prefix sum returns to each element of a sequence the sum of previous elements. The summation or prefix sum needs to be applied using a variety of associative operators, including min, max, and addition.

We refer to all the operations other than sorting as the *basic matrix operation*. The basic matrix operations on m elements can all be implemented with $O(m)$ work and $O(\log m)$ time on the EREW PRAM [19], and with $O(m/B)$ cache complexity and $O(\log m)$ depth in the parallel cache oblivious model. For the parallel cache oblivious model, we assume a tall cache $M > B^2$, where M is the size of the cache and B is the block size. Sorting m elements takes $O(m \log m)$ work and $O(\log m)$ time on an EREW PRAM [9], and $O(\frac{m}{B} \log_{M/B} m)$ cache complexity and $O(\log^2 m)$ depth on the parallel cache oblivious model [3]. All algorithms described in this paper are *cache efficient* in the sense that the cache complexity in the cache oblivious model is bounded by $O(w/B)$ where w is the work in the EREW model. All algorithms use a polylogarithmic number of calls to the basic matrix operations and sorting and are thus in RNC—do

polynomial work with polylogarithmic depth and possibly use randomization.

Given this set up, the problems considered in this paper can be defined as follows:

Facility Location. The goal of this problem is to find a set of facilities $F_S \subseteq F$ that minimizes the objective function

$$\text{FACLOC}(F_S) = \sum_{i \in F_S} f_i + \sum_{j \in C} d(j, F_S) \qquad (1)$$

Note that we do not need an explicit client-to-facility assignment because given a set of facilities F_S, the cost is minimized by assigning each client to the closest open facility.

Non-trivial upper- and lower-bounds for the cost of the optimal solution are useful objects in approximation algorithms. For each client $j \in C$, let $\gamma_j = \min_{i \in F}(f_i + d(j, i))$ and $\gamma = \max_{j \in C} \gamma_j$. The following bounds can be easily established:

$$\gamma \leq \text{opt} \leq \sum_{j \in C} \gamma_j \leq \gamma n_c. \qquad (2)$$

Furthermore, metric facility location has a natural integer-program formulation for which the relaxation yields the pair of primal and dual programs shown in Figure 1.

k-Median and k-Means. Unlike facility location, the k-median objective does not take into consideration facility costs, instead limiting the number of opened centers (facilities) to k. Moreover, in these problems, we typically do not distinguish between facilities and clients; every node is a client, and every node can be a facility. Formally, let $V \subseteq X$ be the set of nodes, and the goal is to find a set of at most k centers $F_S \subseteq V$ that minimizes the objective $\text{KMED}(F_S) = \sum_{j \in V} d(j, F_S)$. Almost identical to k-median is the k-means problem with the objective $\text{KMEANS}(F_S) = \sum_{j \in C} d^2(j, F_S)$.

k-Center. Another type of facility-location problem which has a hard limit on the number of facilities to open is k-center. The k-center problem is to find a set of at most k centers $F_S \subseteq V$ that minimizes the objective $\text{KCENTER}(F_S) = \max_{j \in V} d(j, F_S)$. In these problems, we will use n to denote the size of V.

3. DOMINATOR SET

We introduce and study two variants of the maximal independent set (MIS) problem, which will prove to be useful in nearly all algorithms described in this work. The first variant, called the *dominator set* problem, concerns finding a maximal set $I \subseteq V$ of nodes from a simple graph $G = (V, E)$ such that none of these nodes share a common neighbor (neighboring nodes of G cannot both be selected). The second variant, called the *U-dominator set* problem, involves finding a maximal set $I \subseteq U$ of the U-side nodes of a bipartite graph $H = (U, V, E)$ such that none of the nodes have a common V-side neighbor. We denote by $\text{MAXDOM}(G)$ and $\text{MAXUDOM}(H)$ the solutions to these problems, resp.

Both variants can be equivalently formulated in terms of maximal independent set. The first variant amounts to finding a maximal independent set on

$$G^2 = (V, \{uw : uw \in E \text{ or } \exists z \text{ s.t. } uz, zw \in E\}),$$

and the second variant a maximal independent set on

$$H' = (U, \{uw : \exists z \in V \text{ s.t. } uz, zw \in E\}).$$

Because of this relationship, on the surface, it may seem that one could simply compute G^2 or H' and run an existing MIS algorithm. Unfortunately, computing graphs such as G^2 and H' appears to need $O(n^\omega)$ work, where ω is the matrix-multiply constant, whereas the naïve greedy-like sequential algorithms for the same problems run in $O(|E|) = O(n^2)$. This difference makes it unlikely to obtain work efficient algorithms via this route.

In this section, we develop near work-efficient algorithms for these problems, bypassing the construction of the intermediate graphs. The key idea is to compute a maximal independent set in-place. Numerous parallel algorithms are known for maximal independent set, but the most relevant to us is an algorithm of Luby [25], which we now sketch.

The input to the algorithm is a graph $G = (V, E)$. Luby's algorithm constructs a maximal independent set $I \subseteq V$ by proceeding in multiple rounds, with each round performing the following computation:

Algorithm 3.1 The select step of Luby's algorithm for maximal independent set.

---

1. For each $i \in V$, **in parallel**, $\pi(i) = $ a number chosen u.a.r. from $\{1, 2, \ldots, 2n^4\}$.

2. Include a node i in the maximal independent set I if $\pi(i) < \min\{\pi(j) : j \in \Gamma(i)\}$, where $\Gamma(i)$ is the neighborhood of i in G.

This process is termed the *select step* in Luby's work. Following the select step, the newly selected nodes, together with their neighbors, are removed from the graph before moving on to the next round.

Implementing the select step: We describe how the select step can be performed in-place for the first variant; the technique applies to the other variant. We will be simulating running Luby's algorithm on G^2, without generating G^2. Since G^2 has the same node set as G, step 1 of Algorithm 3.1 remains unchanged. Thus, the crucial computation for the select step is to determine efficiently, for each node i, whether $\pi(i)$ holds the smallest number among its neighbors in G^2, i.e., computing efficiently the test in step 2. To accomplish this, we simply pass the $\pi(i)$ to their neighbors taking a minimum, and then to the neighbors again taking a minimum. These can be implemented with a constant number of basic matrix operations, in particular distribution and summation using minimum over the rows and columns of the $|V|^2$ matrix.

Lemma 3.1 *Given a graph $G = (V, E)$, a maximal dominator set $I \subseteq V$ can be found in expected $O(\log^2 |V|)$ depth and $O(|V|^2 \log |V|)$ work. Furthermore, given a bipartite graph $G = (U, V, E)$, a maximal U-dominator set $I \subseteq U$ can be found in expected $O((\log |U|) \cdot \max\{\log |U|, \log |V|\})$ depth and $O(|V||U| \max\{\log |U|, \log |V|\})$ work.*

For sparse matrices, which we do not use in this paper, this can easily be improved to $O(|E| \log |V|)$ work.

4. FACILITY LOCATION: GREEDY

The greedy scheme underlies an exceptionally simple algorithm for facility location, due to Jain et al. [18]. Despite the simplicity, the algorithm offers one of the best known

Minimize	$\sum_{i\in F, j\in C} d(j,i)x_{ij} \;+\; \sum_{i\in F} f_i y_i$	**Maximize**	$\sum_{j\in C}\;\alpha_j$

Subj. to: $\begin{cases} \sum_{i\in F} x_{ij} & \geq & 1 & \text{for } j\in C \\ y_i - x_{ij} & \geq & 0 & \text{for } i\in F, j\in C \\ x_{ij}\geq 0,\; y_i\geq 0 \end{cases}$
Subj. to: $\begin{cases} \sum_{j\in C}\beta_{ij} & \leq & f_i & \text{for } i\in F \\ \alpha_j - \beta_{ij} & \leq & d(j,i) & \text{for } i\in F, j\in C \\ \beta_{ij}\geq 0,\; \alpha_j\geq 0 \end{cases}$

Figure 1: The primal (left) and dual (right) programs for metric (uncapacitated) facility location.

approximation guarantees for the problem. To describe the algorithm, we will need some definitions.

Definition 4.1 (Star, Price, and Maximal Star) *A star* $\mathcal{S}=(i,C')$ *consists of a facility* i *and a subset* $C'\subseteq C$. *The price of* $\mathcal{S}$ *is* $\mathsf{price}(\mathcal{S})=(f_i+\sum_{j\in C'}d(j,i))/|C'|$. *A star* $\mathcal{S}$ *is said to be* maximal *if all* strict *super sets of* C' *have a larger price, i.e., for all* $C''\supsetneq C'$, $\mathsf{price}((i,C'')) > \mathsf{price}((i,C'))$.

The greedy algorithm of Jain et al. proceeds as follows:

> Until no client remains, pick the cheapest star (i,C'), open the facility i, set $f_i=0$, remove all clients in C' from the instance, and repeat.

This algorithm has a sequential running time of $O(m\log m)$, and using techniques known as factor-revealing LP, Jain et al. show that the algorithm has an approximation factor of 1.861 [18]. From a parallelization point of view, the algorithm is highly sequential—at each step, only the minimum-cost option is chosen, and every subsequent step depends on the preceding one. In this section, we describe how to overcome this sequential nature and obtain an RNC algorithm inspired by the greedy algorithm of Jain et al. We show that the parallel algorithm is a $(6+\varepsilon)$-approximation using elementary techniques. It remains an open problem whether factor-revealing LP techniques can be used to show a better approximation bound for the parallel algorithm.

The key idea to parallelization is that much faster progress will be made if we allow a small slack in what can be selected in each round; however, a subselection step is necessary to ensure that facility and connection costs are properly accounted for.

We present the parallel algorithm in Algorithm 4.1 and now describe step 1 in greater detail; steps 2 – 3 can be implemented using standard techniques [19, 24]. As observed in Jain et al. [18], for each facility i, the lowest-priced star centered at i consists of the κ_i closest clients to i, for some κ_i. Following this observation, we can presort the distance between facilities and clients for each facility. Let i be a facility and assume without loss of generality that $d(i,1) \leq d(i,2) \leq \cdots \leq d(i,n_c)$. Then, the cheapest maximal star for this facility can be found as follows. Use a prefix-sum computation to compute the sequence $p^{(i)} = \{(f_i + \sum_{j\leq k} d(i,k))/k\}_{k=1}^{n_c}$. Then, find the smallest index k such that $p_k^{(i)} < p_{k+1}^{(i)}$ or use $k=n_c$ if no such index exists. It is easy to see that the maximal lowest-priced star centered at i is the facility i together with the client set $\{1,\ldots,k\}$.

Crucial to this algorithm is a subselection step, which ensures that every facility and the clients that connect to it are adequately accounted for in the dual-fitting analysis. This subselection process can be seen as scaling back on the aggressiveness of opening up the facilities, mimicking the greedy algorithm's behavior more closely.

4.1 Analysis

We present a dual-fitting analysis of the above algorithm. The analysis relies on the client-to-facility assignment π, defined in the description of the algorithm. The following easy-to-check facts will be useful in the analysis.

Fact 4.2 *For each iteration of the execution, the following holds: (1) If* $\mathcal{S}_i$ *is the cheapest maximal star centered at* i, *then* j *appears in* $\mathcal{S}_i$ *if and only if* $d(j,i) \leq \mathsf{price}(\mathcal{S}_i)$. *(2) If* $t = \mathsf{price}(\mathcal{S}_i)$, *then* $\sum_{j\in C}\max(0, t-d(j,i)) = f_i$.

Now consider the dual program in Figure 1. For each client j, set α_j to be the τ setting in the iteration that the client was removed. For convenience, assume without loss of generality that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{n_c}$, and define $W_i = \{j \in C : \alpha_j \geq (3+\varepsilon)d(j,i)\}$ for all $i\in F$. We begin the analysis by relating the cost of the solution that the algorithm outputs to the cost of the dual program.

Lemma 4.3 *The cost of the algorithm's solution* $\sum_{i\in F_A} f_i + \sum_{j\in C} d(j,F_A)$ *is upper-bounded by* $2(1+\varepsilon)^2\sum_{j\in C}\alpha_j$.

PROOF. Consider that in step 4(c), a facility i is opened if at least a $\frac{1}{2(1+\varepsilon)}$ fraction of the neighbors "chose" i. Furthermore, we know from the definition of H that, in that round, $f_i + \sum_{j\in\Gamma_H(i)} d(j,i) \leq \tau(1+\varepsilon)\deg(i)$. By noting that we can partition C by which facility the client is assigned to in the assignment π, we establish

$$\sum_{j\in C}\alpha_j\cdot 2(1+\varepsilon)^2 \geq \sum_{i\in F_A}\left(f_i + \sum_{j:\pi_j=i} d(j,i)\right)$$
$$\geq \sum_{i\in F_A} f_i + \sum_{j\in C} d(j,F_A),$$

as desired. $\square$

In the series of claims that follows, we show that when scaled down by a factor of $3+\varepsilon$, the α setting determined above is a dual feasible solution.

Claim 4.4 *Let* $i\in F$. *Assuming* W_i *is non-empty, if* j' *is the lowest-numbered client that connects to* i *(i.e.,* $j' = \min\{j : \pi_j = i\}$, *so it is* ∞ *if* i *is never opened), and* $j_0 = \min\left(j', \arg\min_{j\in W_i}\alpha_j\right)$, *then*

$$\sum_{j\in W_i}\max(0, \alpha_{j_0} - d(j,i)) \;\leq\; f_i.$$

PROOF. Let R be the set of clients which remain at the beginning of the iteration that τ was set to α_{j_0}. First, note that if $j_0 < j'$, then i must be opened after this iteration. As a consequence of Fact 4.2, we have $\sum_{j\in W_i}\max(0, \alpha_{j_0} - d(j,i)) \leq \sum_{j\in R}\max(0, (1+\varepsilon)\alpha_{j_0} - d(j,i)) < f_i$. Therefore,

---

Algorithm 4.1 Parallel greedy algorithm for metric facility location.

In rounds, the algorithm performs the following steps until no client remains:

1. For each facility i, **in parallel**, compute $\mathcal{S}_i = (i, C^{(i)})$, the lowest-priced maximal star centered at i.

2. Let $\tau = \min_{i \in F} \mathsf{price}(\mathcal{S}_i)$, and let $I = \{i \in F : \mathsf{price}(\mathcal{S}_i) \le \tau(1 + \varepsilon)\}$.

3. Construct a bipartite graph $H = (I, C', \{ij : d(i, j) \le \tau(1 + \varepsilon)\})$, where $C' = \{j \in C : \exists i \in I \text{ s.t. } d(i, j) \le \tau(1 + \varepsilon)\}$.

4. **Facility Subselection:** while ($I \ne \emptyset$):

   (a) Let $\Pi : I \to \{1, \ldots, |I|\}$ be a random permutation of I.

   (b) For each $j \in C'$, let $\varphi_j = \arg\min_{i \in \Gamma_H(j)} \Pi(i)$.

   (c) For each $i \in I$, if $|\{j : \varphi_j = i\}| \ge \frac{1}{2(1+\varepsilon)} \deg(i)$, add i to F_A (open i), set $f_i = 0$, remove i from I, and remove $\Gamma_H(i)$ from both C and C'.

   Note: In the analysis, the clients removed in this step have π_j set as follows. If the facility φ_j is opened, let $\pi_j = \varphi_j$; otherwise, π_j is set to any facility i we open in this step such that $ij \in E(H)$. Note that any facility that is opened is at least $1/(2(1+\varepsilon))$ paid for by the clients that select it, and that since every client is assigned to at most one facility, they only pay for one edge.

   (d) Remove $i \in I$ (and the incident edges) from the graph H if on the remaining graph, $\frac{f_i + \sum_{j \in \Gamma_H(i)} d(j,i)}{\deg(i)} > \tau(1 + \varepsilon)$. These facilities will show up in the next round (outer-loop).

   Note: After f_i is set to 0, facility i will still show up in the next round.

---

in the rest of the proof, we will assume that $j_0 = j'$ and that i was actually opened.

Now suppose for a contradiction that $\sum_{j \in W_i} \max(0, \alpha_{j_0} - d(j, i)) > f_i$, so then because $j_0 = j'$, we know that R is in fact the set of clients that remain at the beginning of the iteration in which i was first opened (i.e., i was a center of some star and $f_i > 0$ at that point). Furthermore, we know that if the cost the star centered at i in this iteration is t_i, then $f_i = \sum_{j \in R} \max(0, t_i - d(j, i))$. Therefore, since $W_i \subseteq R$ and $t_i \ge \alpha_{j_0}$, we have $f_i = \sum_{j \in R} \max(0, t_i - d(j, i)) \ge \sum_{j \in R} \max(0, \alpha_{j_0} - d(j, i)) \ge \sum_{j \in W_i} \max(0, \alpha_{j_0} - d(j, i)) > f_i$, which gives a contradiction. $\square$

Claim 4.5 *Let $i \in F$, and j, j' be clients such that $j' \le j$. Then, $\alpha_j \le \alpha_{j'} + d(i, j') + d(i, j)$.*

The proof of this claim closely parallels that of Jain et al. [18] and is omitted due to space considerations. Combining Claims 4.5 and 4.4, we have the following lemma:

Lemma 4.6 *Let $i \in F$. Then,*

$$\sum_{j \in W_i} \left[\alpha_j - (3 + \varepsilon)d(j, i) \right] \le (3 + \varepsilon)f_i.$$

PROOF. We first note that if W_i is empty, the lemma is trivially true. Thus, we assume W_i is non-empty and define j_0 as follows: let j' be the lowest-numbered client that connects to i or ∞ if i is never opened. Then, $j_0 = \min(j', \arg\min_{j \in W_i} \alpha_j)$. Now if $j_0 = j'$, then $d(j_0, i) \le (1 + \varepsilon)\alpha_{j_0}$. Otherwise, we have $j_0 \in W_i$, in which case $d(j_0, i) \le \alpha_j$ by the definition of W_i. Therefore, in either case, $d(j_0, i) \le (1 + \varepsilon)\alpha_{j_0}$.

Next we let $T = \{j \in W_i : \alpha_{j_0} \ge d(j, i)\}$. Applying

Claims 4.5 and 4.4, we have

$$\sum_{j \in W_i} (\alpha_j - d(j, i))$$
$$\le \sum_{j \in W_i} (\alpha_{j_0} + d(j_0, i)) \le \sum_{j \in W_i} (2 + \varepsilon)\alpha_{j_0}$$
$$\le (2 + \varepsilon)f_i + \sum_{j \in T} (2 + \varepsilon)d(j, i) + \sum_{j \in W_i \setminus T} (2 + \varepsilon)d(j, i)$$
$$\le (2 + \varepsilon)f_i + \sum_{j \in W_i} (2 + \varepsilon)d(j, i),$$

which completes the proof. $\square$

Using this lemma, we argue that the setting $\alpha'_j = \alpha_j/(3 + \varepsilon)$ and $\beta'_{ij} = \max(0, \alpha'_j - d(j, i))$ yields a dual feasible solution. First, our choice of β'_{ij}'s ensures that all constraints of the form $\alpha_j - \beta_{ij} \le d(j, i)$ are satisfied. Then, by the lemma above, we have $\sum_{j \in C} \max(0, \alpha_j - (3 + \varepsilon)d(j, i)) = \sum_{j \in W_i} [\alpha_j - (3 + \varepsilon)d(j, i)] \le (3 + \varepsilon)f_i$, which implies that $\sum_{j \in C} \max(0, \alpha_j - (3 + \varepsilon)d(j, i)) \le (3 + \varepsilon)f_i$. Hence, we conclude that for all facility $i \in F$, $\sum_{j \in C} \beta'_{ij} \le f_i$, proving the following corollary:

Corollary 4.7 *The setting $\alpha'_j = \frac{\alpha_j}{3+\varepsilon}$ and $\beta'_{ij} = \max(0, \alpha'_j - d(j, i))$ is a dual feasible solution.*

Running time analysis.

Consider the algorithm's description in Algorithm 4.1. The rows can be presorted to give each client its distances from facilities in order. In the original order, each element can be marked with its rank. Step 1 then involves a prefix sum on the sorted order to determine how far down the order to go and then selection of all facilities at or below that rank. Steps 2–3 require reductions and distributions across the rows or columns of the matrix. The subset $I \subset F$ can be represented as a bit mask over F. Step 4 is more interesting to analyze; the following lemma bounds the number of rounds facility

subselection is executed, the proof of which is analogous to Lemma 4.1.2 of Rajagopalan and Vazirani [33]; we present here for completeness a simplified version of their proof, which suffices for our lemma.

Lemma 4.8 *With probability $1 - o(1)$, the subselection step terminates within $O(\log_{1+\varepsilon} m)$ rounds.*

PROOF. Let $\Phi = |E|$. We will show that if Φ' is the potential value after an iteration of the subselection step, then $\mathbf{E}[\Phi - \Phi'] \geq c\Phi$, for some constant $c > 0$. The lemma then follows from standard results in probability theory. To proceed, define $\mathsf{chosen}_i = |\{j \in C' : \varphi_j = i\}|$. Furthermore, we say that an edge ij is *good* if at most $\theta = \frac{1}{2}(1 - \frac{1}{1+\varepsilon})$ fraction of neighbors of i have degree higher than j.

Consider a good edge ij. We will estimate $\mathbf{E}[\mathsf{chosen}_i | \varphi_j = i]$. Since ij is good, we know that

$$\sum_{j' \in \Gamma_H(i)} \mathbf{1}_{\{\deg(j') \leq \deg(j)\}} \geq (1 - \theta) \deg(i).$$

Therefore, $\mathbf{E}[\mathsf{chosen}_i | \varphi_j = i] \geq \frac{1}{2}(1 - \theta) \deg(i)$, as it can be shown that $\mathbf{Pr}[\varphi_{j'} = i | \varphi_j = i] \geq \frac{1}{2}$ for all $j' \in \Gamma_H(i)$ and $\deg(j') \leq \deg(j)$. By Markov's inequality and realizing that $\mathsf{chosen}_i \leq \deg(i)$, we have

$$\mathbf{Pr}\left[\mathsf{chosen}_i \geq \frac{1}{2(1+\varepsilon)} \deg(i) \;\middle|\; \varphi_j = i\right] = p_0 > 0.$$

Finally, we note that $\mathbf{E}[\Phi - \Phi']$ is at least

$$\sum_{ij \in E} \mathbf{Pr}\left[\varphi_j = i \text{ and } \mathsf{chosen}_i \geq \frac{1}{2(1+\varepsilon)} \deg(i)\right] \cdot \deg(j)$$

$$\geq \sum_{\text{good } ij \in E} \frac{1}{\deg(j)} p_0 \deg(j)$$

$$\geq p_0 \sum_{ij \in E} \mathbf{1}_{\{ij \text{ is good}\}}.$$

Since at least θ fraction of the edges are good, $\mathbf{E}[\Phi - \Phi'] \geq p_0 \theta \Phi$. Since $\ln(1/(1 - p_0\theta)) = \Omega(\log(1 + \varepsilon))$, the lemma follows from standard results in probability [30]. $\square$

It is easy to see that each subselection step can be performed with a constant number of basic matrix operations over the D matrix. Therefore, if the number of rounds the main body is executed is r, the algorithm makes $O(r \log_{1+\varepsilon} m)$ calls to the basic matrix operations described in Section 2 with probability exceeding $1 - o(1)$. It also requires a single sort in the preprocessing. This means $O(r \log_{1+\varepsilon} m \log m)$ time implies a total of $O(rm \log_{1+\varepsilon} m)$ work (with probability exceeding $1 - o(1)$) on the EREW PRAM. Furthermore, it is cache efficient (cache complexity is $O(w/B)$) since the sort is only applied once and does not dominate the cache bounds.

Bounding the number of rounds.

Before describing a less restrictive alternative, we point out that the simplest way to bound the number of rounds by a polylogarithm factor is to rely on the common assumption that the facility cost, as well as the ratio between the minimum (non-zero) and the maximum client-facility distance, is polynomially bounded in the input size. As a result of this assumption, the number of rounds is upper-bounded by $\log_{1+\varepsilon}(m^c) = O(\log_{1+\varepsilon} m)$, for some $c \geq 1$.

Alternatively, we can apply a preprocessing step to ensure that the number of rounds is polylogarithm in m. The basic idea of the preprocessing step is that if a star is "relatively cheap," opening it right away will harm the approximation factor only slightly. Using the bounds in Equation (2), if $\mathcal{S}_i$ is the lowest-priced maximal star centered at i, we know we can afford to open i and discard all clients attached to it if $\mathsf{price}(\mathcal{S}_i) \leq \frac{\gamma}{m^2}$. Therefore, the preprocessing step involves: (1) computing $\mathcal{S}_i$, the lowest-priced maximal star centered at i, for all $i \in F$, (2) opening all i such that $\mathsf{price}(\mathcal{S}_i) \leq \frac{\gamma}{m^2}$, (3) setting f_i of these facilities to 0 and removing all clients attached to these facilities.

Computing γ takes $O(\log n_c + \log n_f)$ depth and $O(m)$ work. The rest of the preprocessing step is at most as costly as a step in the main body. Thus, the whole preprocessing step can be accomplished in $O(\log m)$ depth and $O(m)$ work. With this preprocessing step, three things are clear: First, τ in the first iteration of the main algorithm will be at least $\frac{\gamma}{m^2}$, because cheaper stars have already been processed in preprocessing. Second, the cost of our final solution is increased by at most $n_c \times \frac{\gamma}{m^2} \leq \frac{\gamma}{m} \leq \mathsf{opt}/m$, because the facilities and clients handled in preprocessing can be accounted for by the cost of their corresponding stars—specifically, there can be most n_c stars handled in preprocessing, each of which has price $\leq \gamma/m^2$; and the price for a star includes both the facility cost and the connection cost of the relevant clients and facilities. Finally, in the final iteration, $\tau \leq n_c \gamma$. As a direct consequence of these observations, the number of rounds is upper-bounded by $\log_{1+\varepsilon}(\frac{n_c \gamma}{\gamma/m^2}) \leq \log_{1+\varepsilon}(m^3) = O(\log_{1+\varepsilon} m)$, culminating in the following theorem:

Theorem 4.9 *Let $0 < \varepsilon \leq 1$ be fixed. For sufficiently large input, there is a greedy-style RNC $O(m \log_{1+\varepsilon}^2(m))$-work algorithm that yields a factor-$(6 + \varepsilon)$ approximation for the metric facility-location problem.*

5. FACILITY LOCATION: PRIMAL-DUAL

The primal-dual scheme is a versatile paradigm for combinatorial algorithms design. In the context of facility location, this scheme underlies the Lagrangian-multiplier preserving[1] (LMP) 3-approximation algorithm of Jain and Vazirani, enabling them to use the algorithm as a subroutine in their 6-approximation algorithm for k-median [17].

The algorithm of Jain and Vazirani consists of two phases, a primal-dual phase and a postprocessing phase. To summarize this algorithm, consider the primal and dual programs in Figure 1. In the primal-dual phase, starting with all dual variables set to 0, we raise the dual variables α_j's uniformly until a constraint of the form $\alpha_j - \beta_{ij} \leq d(j, i)$ becomes tight, at which point β_{ij} will also be raised, again, uniformly to prevent these constraints from becoming overtight. When a constraint $\sum_j \beta_{ij} \leq f_i$ is tight, facility i is tentatively opened and clients with $\alpha_j \geq d(j, i)$ are "frozen," i.e., we stop raising their α_j values from this point on. The first phase ends when all clients are frozen. In the postprocessing phase, we compute and output a maximal independent set on a graph G of tentatively open facilities; in this graph, there is an edge between a pair of facilities i and i' if there is a client j such that $\alpha_j > d(j, i)$ and $\alpha_j > d(j, i')$. Thus, the maximal

---

[1]This means $\alpha \sum_{i \in F_A} f_i + \sum_{j \in C} d(j, F_A) \leq \alpha \cdot \mathsf{opt}$, where α is the approximation ratio.

independent set ensures proper accounting of the facility cost (i.e., each client "contributes" to at most one open facility, and every open facility has enough contribution). Informally, we say that a client j "pays" for or "contributes" to a facility i if $\beta_{ij} = \alpha_j - d(j, i) > 0$.

Remarks. We note that in the parallel setting, the description of the postprocessing step above does not directly lead to an efficient algorithm, because constructing G in polylogarithmic depth seems to need $O(mn_f)$ work, which is much more than one needs sequentially.

In this section, we show how to obtain a work-efficient RNC $(3+\varepsilon)$-approximation algorithm for facility location, based on the primal-dual algorithm of Jain and Vazirani. Critical to bounding the number of iterations in the main algorithm by $O(\log m)$ is a preprocessing step, which is similar to that used by Pandit and Pemmaraju in their distributed algorithm [32].

Preprocessing: Assuming γ as defined in Equation (2), we will open every facility i that satisfies

$$\sum_{j \in C} \max\left(0, \frac{\gamma}{m^2} - d(j,i)\right) \geq f_i.$$

Furthermore, for all clients j such that there exists an opened i and $d(j,i) \leq \gamma/m^2$, we declare them connected and set $\alpha_j = 0$. The facilities opened in this step will be called *free* facilities and denoted by the set F_0.

Main Algorithm: The main body of the algorithm is described in Algorithm 5.1. The algorithm outputs a bipartite graph $H = (F_T, C, E)$, constructed as the algorithm executes. Here F_T is the set of facilities declared open during the iterations of the main algorithm and E is given by $E = \{ij : i \in F, j \in C, \text{ and } (1+\varepsilon)\alpha_j > d(j,i)\}$.

Post-processing. As a post-processing step, we compute $I = \text{MAxUDom}(H)$. Thus, the set of facilities $I \subseteq F_T$ has the property that each client contributes to the cost of at most one facility in I. Finally, we report $F_A = I \cup F_0$ as the set of facilities in the final solution.

5.1 Analysis

To analyze approximation guarantee of this algorithm, we start by establishing that the α_j setting produced by the algorithm leads to a dual feasible solution.

Claim 5.1 *For any facility i,*

$$\sum_{j \in \Gamma_H(i)} \max(0, \alpha_j - d(j,i)) \leq f_i.$$

The proof of this claim is omitted in the interest of space. It follows from this claim that setting $\beta_{ij} = \max(0, \alpha_j - d(j,i))$ provides a dual feasible solution. Next we relate the cost of our solution to the cost of the dual solution. To ease the following analyses, we use a client-to-facility assignment $\pi \colon C \to F$, defined as follows: For all $j \in C$, let $\varphi(j) = \{i : (1+\varepsilon)\alpha_j \geq d(j,i)\}$. Now for each client j, **(1)** if there exists $i \in F_0$ such that $d(j,i) \leq \gamma/m^2$, set π_j to *any* such i; **(2)** if there exists $i \in I$ such that ij is an edge in H, then $\pi_j = i$ (i is unique because of properties of I); **(3)** if there exists $i \in I$ such that $i \in \varphi(j)$, then $\pi_j = i$; **(4)** otherwise, pick $i' \in \varphi(j)$ and set π_j to $i \in I$ which is a neighbor of a neighbor of i'.

Clients of the first case, denoted by C_0, are called *freely connected*; clients of the cases (2) and (3), denoted by C_1,

are called *directly connected*. Otherwise, a client is *indirectly connected*.

The following lemmas bound the facility costs and the connection costs of indirectly connected clients.

Lemma 5.2

$$\sum_{i \in F_A} f_i \leq \frac{\gamma}{m} + \sum_{j \in C_1}(1+\varepsilon)\alpha_j - \sum_{j \in C_0 \cup C_1} d(j, \pi_j)$$

PROOF. When facility $i \in F_T$ was opened, it must satisfy $f_i \leq \sum_{j : ij \in E(G)}(1+\varepsilon)\alpha_j - d(j,i)$. If client j has contributed to i (i.e., $(1+\varepsilon)\alpha_j - d(j,i) > 0$) and $i \in I$, then j is directly connected to it. Furthermore, for each client j, there is at most one facility in I that it contributes to (because $I = \text{MAxUDom}(H)$). Therefore, $\sum_{i \in I} f_i \leq \sum_{j \in C_1}(1+\varepsilon)\alpha_j - d(j, \pi_j)$. Furthermore, for each "free" facility, we know that $f_i \leq \sum_{j \in C} \max(0, \gamma^2/m^2 - d(j,i))$, so by our choice of π, $f_i \leq \frac{\gamma}{m^2} \times n_c - \sum_{j \in C_0 : \pi_j = i} d(j,i)$. Thus, $\sum_{i \in F_0} f_i \leq \gamma/m - \sum_{j \in C_0} d(j,i)$. Combining these results and observing that F_A is the disjoint union of I and F_0, we have the lemma. $\square$

Lemma 5.3 *For each indirectly connected client j (i.e., $j \notin C_0 \cup C_1$), we have $d(j, \pi_j) \leq 3(1+\varepsilon)\alpha_j$.*

PROOF. Because $j \notin C_0 \cup C_1$ and $I = \text{MAxUDom}(H)$, there must exist a facility $i' \in \varphi(j)$ and a client j' such that j' contributed to both i and i', and $(1+\varepsilon)\alpha_j \geq d(j,i')$. We claim that both $d(j',i')$ and $d(j',i)$ are upper-bounded by $(1+\varepsilon)\alpha_j$. To see this, we note that because j' contributed to both i and i', $d(j',i') \leq (1+\varepsilon)\alpha_{j'}$ and $d(j',i) \leq (1+\varepsilon)\alpha_{j'}$. Let ℓ be the iteration in which j was declared frozen, so $\alpha_j = t_\ell$. Since $i' \in \varphi(j)$, i' must be declared open in iteration $\leq \ell$. Furthermore, because $(1+\varepsilon)\alpha_j > d(j',i')$, $\alpha_{j'}$ must be frozen in or prior to iteration ℓ. Consequently, we have $\alpha_{j'} \leq t_\ell = \alpha_j$. Combining these facts and applying the triangle inequality, we get $d(j,i) \leq d(j,i') + d(i',j') + d(j',i) \leq (1+\varepsilon)\alpha_j + 2(1+\varepsilon)\alpha_{j'} \leq 3(1+\varepsilon)\alpha_j$. $\square$

By Lemmas 5.2 and 5.3, we establish

$$3\sum_{i \in F_A} f_i + \sum_{j \in C} d(j, \pi_j) \leq \frac{3\gamma}{m} + 3(1+\varepsilon)\sum_{j \in C} \alpha_j. \quad (3)$$

Now since $\{\alpha_j, \beta_{ij}\}$ is dual feasible, its value can be at most that of the primal optimal solution; that is, $\sum_j \alpha_j \leq \mathsf{opt}$. Therefore, combining with Equation (3), we know that the cost of the solution returned by parallel primal-dual algorithm in this section is at most $3\sum_{i \in F_A} f_i + \sum_{j \in C} d(j, C) \leq (3 + \varepsilon')\mathsf{opt}$ for some $\varepsilon' > 0$ when the problem instance is large enough.

Running Time Analysis.

We analyze the running of the algorithm presented, starting with the main body of the algorithm. Since $\sum_j \alpha_j \leq \mathsf{opt}$ and $\mathsf{opt} \leq n_c\gamma$, no α_j can be bigger than $n_c\gamma \leq m\gamma$. Hence, the main algorithm must terminate before $\ell > 3\log_{1+\varepsilon} m$, which upper-bounds the number of iterations to $O(\log_{1+\varepsilon} m)$. In each iteration, steps 1, 3, and 4 perform trivial work. Step 2 can be broken down into (1) computing the max for all $i \in F, j \in C$, and (2) computing the sum for each $i \in F$. These can all be implemented with the basic matrix operations, giving a total of $O(\log_{1+\varepsilon} m)$ of basic matrix operations over a matrix of size m.

Algorithm 5.1 Parallel primal-dual algorithm for metric facility location

---

For iteration $\ell = 0, 1, \ldots$, the algorithm performs the following steps until all facilities are opened or all clients are frozen, whichever happens first.

1. For each unfrozen client j, **in parallel**, set α_j to $\frac{\gamma}{m^2}(1 + \varepsilon)^\ell$.

2. For each unopened facility i, **in parallel**, declare it open if

$$\sum_{j \in C} \max(0, (1 + \varepsilon)\alpha_j - d(j, i)) \geq f_i.$$

3. For each unfrozen client j, **in parallel**, freeze this client if there exists an opened facility i such that $(1 + \varepsilon)\alpha_j \geq d(j, i)$.

4. Update the graph H by adding edges between pairs of nodes ij such that $(1 + \varepsilon)\alpha_j > d(j, i)$.

After the last iteration, if all facilities are opened but some clients are *not* yet frozen, we determine in parallel the α_j settings of these clients that will make them reach an open facility (i.e., $\alpha_j = \min_i d(j, i)$). Finally, update the graph H as necessary.

The preprocessing step, again, involves some reductions over the rows and columns of the matrix. This includes the calculations of γ_j's and the composite γ. The post-processing step relies on computing the U-dominating set, as described in Section 2 which runs in $O(\log m)$ matrix operations.

The whole algorithm therefore runs in $O(\log_{1+\varepsilon} m)$ basic matrix operations and is hence work efficient compared to the $O(m \log m)$ sequential algorithm of Jain and Vazirani. Putting these altogether, we have the following theorem:

Theorem 5.4 *Let $\varepsilon > 0$ be fixed. For sufficiently large m, there is a primal-dual* RNC *$O(m \log_{1+\varepsilon} m)$-work algorithm that yields a factor-$(3 + \varepsilon)$ approximation for the metric facility-location problem.*

6. OTHER RESULTS

In this section, we consider other applications of dominator set in facility-location problems.

6.1 k-Center

Hochbaum and Shmoys [16] show a simple factor-2 approximation for k-center. The algorithm performs a binary search on the range of distances. We show how to combine the dominator-set algorithm from Section 3 with standard techniques to parallelize the algorithm of Hochbaum and Shmoys, resulting in an RNC algorithm with the same approximation guarantee. Consider the set of distances $\mathcal{D} = \{d(i, j) : i \in C \text{ and } j \in V\}$ and order them so that $d_1 < d_2 < \cdots < d_p$ and $\{d_1, \ldots, d_p\} = \mathcal{D}$, where $p = |\mathcal{D}|$. The sequence $\{d_i\}_{i=1}^p$ can be computed in $O(\log |V|)$ depth and $O(|V|^2 \log |V|)$ work. Let H_α be a graph defined as follows: the nodes of H_α is the set of nodes V, but there is an edge connecting i and j if and only if $d(i, j) \leq \alpha$.

The main idea of the algorithm is simple: find the smallest index $t \in \{1, 2, \ldots, p\}$ such that $\text{MAXDOM}(H_{d_t}) \leq k$. Hochbaum and Shmoys observe that the value t can be found using binary search in $O(\log p) = O(\log |V|)$ probes. We parallelize the probe step, consisting of constructing $H_{d_{t'}}$ for a given $t' \in \{1, \ldots, p\}$ and checking whether $|\text{MAXDOM}(H_{d_{t'}})|$ is bigger than k. Constructing $H_{d_{t'}}$ takes $O(1)$ depth and $O(|V|^2)$ work, and using the maximal-dominator-set algorithm from Section 3, the test can be performed in expected $O(\log^2 |V|)$ depth and expected $O(|V|^2 \log |V|)$ work. The approximation factor is identical to the original algorithm, hence proving the following theorem:

Theorem 6.1 *There is an* RNC *2-approximation algorithm with $O((|V| \log |V|)^2)$ work for k-center.*

6.2 Facility Location: LP Rounding

LP rounding was among the very first techniques that yield non-trivial approximation guarantees for metric facility location. The first constant-approximation algorithm was given by Shmoys et al. [34]. Although we do not know how to solve the linear program for facility location in polylogarithmic depth, we demonstrate another application of the dominator-set algorithm and the slack idea by parallelizing the randomized-rounding step of Shmoys et al. The algorithm yields a $(4 + \varepsilon)$-approximation, and the randomized rounding is an RNC algorithm.

The randomized rounding algorithm of Shmoys et al. consists of two phases: a filtering phase and a rounding phase. In the following, we show how to parallelize these phases and prove that the parallel version has a similar guarantee. Our presentation differs slightly from the original work but works in the same spirit.

Filtering: The filtering phase is naturally parallelizable. Fix α to be a value between 0 and 1. Given an optimal primal solution (x, y), the goal of this step is to produce a new solution (x', y') with properties as detailed in Lemma 6.2. Let $\delta_j = \sum_{i \in F} d(i, j) \cdot x_{ij}$, $B_j = \{i \in F : d(i, j) \leq (1 + \alpha)\delta_j\}$, and $\text{mass}(B_j) = \sum_{i \in B_j} x_{ij}$. We compute x'_{ij} and y'_i as follows: (1) let $x'_{ij} = x_{ij}/\text{mass}(B_j)$ if $i \in B_j$ or 0 otherwise, and (2) let $y'_i = \min(1, (1 + 1/\alpha)y_i)$.

Lemma 6.2 *Given an optimal primal solution (x, y), there is a primal feasible solution (x', y') such that (1) $\sum_i x'_{ij} = 1$, (2) if $x'_{ij} > 0$, then $d(j, i) \leq (1 + \alpha)\delta_j$, and (3) $\sum_i f_i y_i \leq (1 + \frac{1}{\alpha})\sum_i f_i y'_i$.*

PROOF. By construction, (1) clearly holds. Furthermore, we know that if $x'_{ij} > 0$, it must be the case that $i \in B_j$, so $d(j, i) \leq (1 + \alpha)\delta_j$, proving (2). By definition of y'_i, $\sum_i f_i y_i \leq (1 + \frac{1}{\alpha})\sum_i f_i y'_i$, proving (3). Finally, since in an optimal LP solution, $\sum_i x_{ij} = 1$, we know that $\text{mass}(B_j) \geq \frac{\alpha}{1+\alpha}$, by an averaging argument. Therefore, $x'_{ij} \leq (1 + \frac{1}{\alpha})x_{ij} \leq \min(1, (1 + \frac{1}{\alpha})y_i) = y'_i$, showing that (x', y') is primal feasible. $\square$

Rounding: The rounding phase is more challenging to parallelize because it is inherently sequential—a greedy algorithm which considers the clients in an increasing order of δ_j and

appears to need $\Omega(n_c)$ steps. We show, however, that we can achieve parallelism by eagerly processing the clients $S = \{j : \delta_j \leq (1+\varepsilon)\tau\}$. This is followed by a clean-up step, which uses the dominator-set algorithm to rectify the excess facilities. We precompute the following information: (1) for each j, let i_j be the least costly facility in B_j, and (2) construct $H = (C, F, ij \in E$ iff. $i \in B_j)$.

There is a preprocessing step to ensure that the number of rounds is polylogarithmic in m. Let θ be the value of the optimal LP solution. By an argument similar to that of Section 4, we can afford to process all clients with $\delta_j \leq \theta/m^2$ in the first round, increasing the final cost by at most $\theta/m \leq \mathsf{opt}/m$. The algorithm then proceeds in rounds, each performing the following steps:

---

1. Let $\tau = \min_j \delta_j$.

2. Let $S = \{j : \delta_j \leq (1+\varepsilon)\tau\}$ and

3. Let $J = \mathrm{MAXUDOM}(H)$, add $I = \{i_j : j \in J\}$ to F_A; finally, remove all of S and $\cup_{j \in S} B_j$ from $V(H)$.

Since J is U-dominator of H, we know that for all distinct $j, j' \in J$, $B_j \cap B_{j'} = \emptyset$; therefore, $\sum_{i \in I} f_i = \sum_{j \in J} f_{i_j} \leq \sum_{j \in J} \left(\sum_{i \in B_j} x'_{ij} f_{i_j} \right) \leq \sum_{j \in J} y'_i f_{i_j} \leq \sum_{j \in J} y'_i f_i$, proving the following claim:

Claim 6.3 *In each round, $\sum_{i \in I} f_i \leq \sum_{i \in \cup_j B_j} y'_i f_i$.*

Like our previous analyses, we will define a client-to-facility assignment π convenient for the proof. For each $j \in C$, if $i_j \in F_A$, let $\pi_j = i_j$; otherwise, set $\pi_j = i_{j'}$, where j' is the client that causes i_j to be shut down (i.e., either $i_j \in B_{j'}$ and j' was process in a previous iteration, or both j and j' are processed in the same iteration but there exists $i \in B_j \cap B_{j'}$).

Claim 6.4 *Let j be a client. If $i_j \in F_A$, then $d(j, \pi_j) \leq (1+\alpha)\delta_j$; otherwise, $d(j, \pi_j) \leq 3(1+\alpha)(1+\varepsilon)\delta_j$.*

PROOF. If $i_j \in F_A$, then by Lemma 6.2, $d(j, \pi_j) \leq (1+\alpha)\delta_j$. If $i_j \notin F_A$, we know that there must exist $i \in B_j$ and j' such that $i \in B_{j'}$ and $\delta_{j'} \leq (1+\varepsilon)\delta_j$. Thus, applying Lemma 6.2 and the triangle inequality, we have $d(j, \pi_j) \leq d(j, i) + d(i, j') + d(j', i_{j'}) \leq 3(1+\alpha)(1+\varepsilon)\delta_j$. $\square$

Running Time Analysis: The above algorithm will terminate in at most $O(\log_{1+\varepsilon} m)$ rounds because the preprocessing step ensures the ratio between the maximum and the minimum δ_j values are polynomially bounded. Like previous analyses, steps $1 - 2$ can be accomplished in $O(1)$ basic matrix operations, and step 3 in $O(\log m)$ basic matrix operations on matrices of size m. This yields a total of $O(\log_{1+\varepsilon} m \log m)$ basic matrix operations, proving the following theorem:

Theorem 6.5 *Given an optimal LP solution for the primal LP in Figure 1, there is an* NC *rounding algorithm yielding a $(4+\varepsilon)$-approximation with $O(m \log m \log_{1+\varepsilon} m)$ work. It is cache efficient.*

7. k-MEDIAN: LOCAL SEARCH

Local search, LP rounding, and Lagrangian relaxation are among the main techniques for approximation algorithms for k-median. In this section, building on the algorithms from previous sections, we present an algorithm for the k-median problem, based on local-search techniques. The natural local-search algorithm for k-median is very simple: starting with any set F_A of k facilities, find some $i \in F_A$ and $i' \in F \setminus F_A$ such that swapping them decreases the k-median cost, and repeat until no such moves can be found. Finding an improving swap or identifying that none exists takes $O(k(n-k)n)$ time sequentially, where n is the number of nodes in the instance. This algorithm is known to be a 5-approximation [1, 14].

The key ideas in this section are that we can find a good initial solution S_0 quickly and perform each local-search step fast. Together, this means that only a small number of local-search steps is needed, and each step can be performed fast. To find a good initial solution, we observe that any optimal k-center solution is an n-approximation for k-median. Therefore, we will use the 2-approximation from Section 6.1 as a factor-$(2n)$ solution for the k-median problem. At the beginning of the algorithm, for each $j \in V$, we order the facilities by their distance from j, taking $O(n^2 \log n)$ work and $O(\log n)$ depth.

Let $0 < \varepsilon < 1$ be fixed. We say that a swap (i, i') such that $i \in F_A$ and $i' \in F \setminus F_A$ is *improving* if $\mathrm{KMED}(F_S - i + i') < (1 - \beta/k)\mathrm{KMED}(F_S)$, where $\beta = \varepsilon/(1+\varepsilon)$. The parallel algorithm proceeds as follows. In each round, find and apply an improving swap as long as there is one. We now describe how to perform each local-search step fast. During the execution, the algorithm keeps track of φ_j, the facility client j is assigned to, for all $j \in V$. We will consider all possible test swaps $i \in F_A$ and $i' \in V \setminus F_A$ *simultaneously in parallel*. For each potential swap (i, i'), every client can independently compute $\Delta_j = d(j, F_A - i + i') - d(j, F_A)$; this computation trivially takes $O(n_c)$ work and $O(1)$ depth, since we know φ_j and the distances are presorted. From here, we know that $\mathrm{KMED}(F_A - i + i') - \mathrm{KMED}(F_A) = \sum_j \Delta_j$, which can be computed in $O(n)$ work and $O(\log n)$ depth. Therefore, in $O(k(n-k)n)$ work and $O(\log n)$ depth, we can find an improving swap or detect that none exists. Finally, a round concludes by applying an improving swap to F_A and updating the φ_j values.

Arya et al. [1] show that the number of rounds is bounded by

$$O\left(\log_{1/(1-\beta/k)}\left(\mathrm{KMED}(S_0)/\mathsf{opt}\right)\right) = O\left(\log_{1/(1-\beta/k)}(n)\right)$$

Since for $0 < \varepsilon < 1$, $\ln\left(1/(1-\beta/k)\right) \leq \frac{2}{k}\ln\left(1/(1-\beta)\right)$, we have the following theorem, assuming constant k, which is often the case in many applications:

Theorem 7.1 *For constant k, there is an* NC *$O(k^2(n-k)n \log_{1+\varepsilon}(n))$-work algorithm which gives a factor-$(5+\varepsilon)$ approximation for k-median.*

Remarks. Relative to the sequential algorithm, this algorithm is work efficient—regardless of the range of k. In addition to k-median, this approach is applicable to k-means, yielding an $(81+\varepsilon)$-approximation [14] in general metric spaces and a $(25+\varepsilon)$-approximation for the Euclidean space [20], and the same parallelization techniques can be used to achieve the same running time. Furthermore, there is a factor-3 approximation local-search algorithm for facility location, in which a similar idea can be used to perform each local-search

step efficiently; however, we do not know how to bound the number of rounds.

8. CONCLUSION

This paper studies the design and analysis of parallel approximation algorithms for facility-location problems, including facility location, k-center, k-median, and k-means. We presented several efficient algorithms, based on a diverse set of approximation algorithms techniques.

References

[1] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[2] B. Berger, J. Rompel, and P. W. Shor. Efficient *NC* algorithms for set cover with applications to learning and geometry. In *FOCS'89*, pages 54–59, 1989.

[3] G. E. Blelloch, P. B. Gibbons, and H. V. Simhadri. Low depth cache-oblivious sorting. In *SPAA'10*, 2010.

[4] J. Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *APPROX'07*, 2007. 29–43.

[5] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.*, 34(4):803–824, 2005. ISSN 0097-5397.

[6] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *J. Comput. System Sci.*, 65(1):129–149, 2002. ISSN 0022-0000. Special issue on STOC, 1999 (Atlanta, GA).

[7] F. A. Chudak. Improved approximation algorithms for uncapacitated facility location. In *Integer programming and combinatorial optimization (IPCO)*, volume 1412 of *Lecture Notes in Comput. Sci.*, pages 180–194. Springer, Berlin, 1998.

[8] F. A. Chudak and D. B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2003. ISSN 0097-5397.

[9] R. Cole. Parallel merge sort. *SIAM J. Comput.*, 17(4):770–785, 1988.

[10] M. Elkin. Distributed approximation: a survey. *SIGACT News*, 35(4):40–57, 2004.

[11] J. Gehweiler, C. Lammersen, and C. Sohler. A distributed $O(1)$-approximation algorithm for the uniform facility location problem. In *SPAA'06*, pages 237–243, 2006.

[12] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38(2-3):293–306, 1985. ISSN 0304-3975.

[13] S. Guha and S. Khuller. Greedy strikes back: improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999. ISSN 0196-6774.

[14] A. Gupta and K. Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

[15] D. S. Hochbaum and D. B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. Assoc. Comput. Mach.*, 33(3):533–550, 1986. ISSN 0004-5411.

[16] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. ISSN 0364765X. URL http://www.jstor.org/stable/3689371.

[17] K. Jain and V. V. Vazirani. 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.

[18] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.

[19] J. JáJá. *An Introduction to Parallel Algorithms*. Addison-Wesley, 1992. ISBN 0-201-54856-9.

[20] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004. ISSN 0925-7721.

[21] S. Khuller, U. Vishkin, and N. E. Young. A primal-dual parallel approximation technique applied to weighted set and vertex covers. *J. Algorithms*, 17(2):280–289, 1994.

[22] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *J. Algorithms*, 37(1):146–188, 2000. ISSN 0196-6774. (Preliminary version in 9th SODA, 1998).

[23] C. Koufogiannakis and N. E. Young. Distributed and parallel algorithms for weighted vertex cover and other covering problems. In *PODC*, pages 171–179, 2009.

[24] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Array, Trees, Hypercubes*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992. ISBN 1-55860-117-1.

[25] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[26] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *STOC'93*, pages 448–457, 1993.

[27] M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. A greedy facility location algorithm analyzed using dual fitting. In *Approximation, randomization, and combinatorial optimization (Berkeley, CA, 2001)*, volume 2129 of *Lecture Notes in Comput. Sci.*, pages 127–137. Springer, Berlin, 2001.

[28] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In *Approximation algorithms for combinatorial optimization*, volume 2462 of *Lecture Notes in Comput. Sci.*, pages 229–242. Springer, Berlin, 2002.

[29] T. Moscibroda and R. Wattenhofer. Facility location: distributed approximation. In *PODC'05*, pages 108–117, 2005.

[30] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995. ISBN 0-521-47465-5.

[31] M. Pál and E. Tardos. Group strategyproof mechanisms via primal-dual algorithms. In *FOCS'03*, pages 584–593, 2003.

[32] S. Pandit and S. V. Pemmaraju. Return of the primal-dual: distributed metric facility location. In *PODC'09*, pages 180–189, 2009.

[33] S. Rajagopalan and V. V. Vazirani. Primal-dual *RNC* approximation algorithms for set cover and covering integer programs. *SIAM J. Comput.*, 28(2):525–540, 1998.

[34] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *STOC*, pages 265–274, 1997.

[35] A. Srinivasan. New approaches to covering and packing problems. In *SODA*, pages 567–576, 2001.

[36] M. Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Integer programming and combinatorial optimization*, volume 2337 of *Lecture Notes in Comput. Sci.*, pages 240–257. Springer, Berlin, 2002.

[37] Q. Wang and K. H. Cheng. Parallel time complexity of a heuristic algorithm for the k-center problem with usage weights. In *Proc. IEEE Symposium on Parallel and Distributed Processing*, pages 254 –257, Dec. 1990.

[38] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS*, pages 538–546, 2001.