Large-Scale Distributed Algorithms for Facility Location with Outliers

Tanmay Inamdar
Department of Computer Science, The University of Iowa, Iowa, USA
tanmay-inamdar@uiowa.edu

Shreyas Pai
Department of Computer Science, The University of Iowa, Iowa, USA
shreyas-pai@uiowa.edu

Sriram V. Pemmaraju
Department of Computer Science, The University of Iowa, Iowa, USA
sriram-pemmaraju@uiowa.edu

Abstract

This paper presents fast, distributed, $O(1)$-approximation algorithms for metric facility location problems with outliers in the Congested Clique model, Massively Parallel Computation (MPC) model, and in the k-machine model. The paper considers *Robust Facility Location* and *Facility Location with Penalties*, two versions of the facility location problem with outliers proposed by Charikar et al. (SODA 2001). The paper also considers two alternatives for specifying the input: the input metric can be provided explicitly (as an $n \times n$ matrix distributed among the machines) or implicitly as the shortest path metric of a given edge-weighted graph. The results in the paper are:

- **Implicit metric**: For both problems, $O(1)$-approximation algorithms running in $O(\text{poly}(\log n))$ rounds in the Congested Clique and the MPC model and $O(1)$-approximation algorithms running in $\tilde{O}(n/k)$ rounds in the k-machine model.
- **Explicit metric**: For both problems, $O(1)$-approximation algorithms running in $O(\log \log \log n)$ rounds in the Congested Clique and the MPC model and $O(1)$-approximation algorithms running in $\tilde{O}(n/k)$ rounds in the k-machine model.

Our main contribution is to show the existence of Mettu-Plaxton-style $O(1)$-approximation algorithms for both Facility Location with outlier problems. As shown in our previous work (Berns et al., ICALP 2012, Bandyapadhyay et al., ICDCN 2018) Mettu-Plaxton style algorithms are more easily amenable to being implemented efficiently in distributed and large-scale models of computation.

2012 ACM Subject Classification Theory of computation → Facility location and clustering, Theory of computation → Distributed algorithms, Theory of computation → MapReduce algorithms, Theory of computation → Graph algorithms analysis

Keywords and phrases Distributed Algorithms, Clustering with Outliers, Metric Facility Location, Massively Parallel Computation, k-machine model, Congested Clique

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2018.5

Related Version [24], `http://arxiv.org/abs/1811.06494`

1 Introduction

Metric Facility Location (in short, FacLoc) is a well-known combinatorial optimization problem used to model clustering problems. The input to the problem is a set F of *facilities*, an *opening cost* $f_i \geq 0$ for each facility $i \in F$, a set C of *clients*, and a metric space $(F \cup C, d)$ of *connection costs*, where $d(i,j)$ denotes the cost of client j connecting to facility i. The objective is to find a subset $F' \subseteq F$ of facilities to open so that the total cost of opening the facilities plus the cost of connecting all clients to open facilities is minimized. In other words, the quantity $cost(F') := \sum_{i \in F'} f_i + \sum_{j \in C} d(j, F')$ is minimized, where $d(j, F')$ denotes $\min_{i \in F'} d(i,j)$. FacLoc is NP-complete, but researchers have devised a number of approximation algorithms for the problem. For any $\alpha \geq 1$, an *α-approximation algorithm* for FacLoc finds in polynomial time, a subset $F' \subseteq F$ of facilities such that $cost(F') \leq \alpha \cdot cost(F^*)$, where F^* is an optimal solution to the given instance of FacLoc. There are several well-known $O(1)$-factor approximation algorithms for FacLoc including the primal-dual algorithm of Jain and Vazirani [25] and the greedy algorithm of Mettu and Plaxton [33]. The best approximation factor currently achieved by an algorithm for FacLoc is 1.488 [30]. More recently, motivated by the need to solve FacLoc and other clustering problems on extremely large inputs, researchers have proposed distributed and parallel approximation algorithms for these problems. See for example [14, 15] for clustering algorithms in systems such as MapReduce [11] and Pregel [32] and [4] for clustering algorithms in the *k-machine model.* Clustering algorithms [38] have also been designed for streaming models of computation [1].

Outliers can pose a problem for many statistical methods. For clustering problems, a few outliers can have an outsized influence on the optimal solution, forcing the opening of costly extra facilities or leading to poorer service to many clients. Versions of FacLoc that are robust to outliers have been proposed by Charikar et al. [10], where the authors also present $O(1)$-approximation algorithms for these problems. Specifically, Charikar et al. [10] propose two versions of FacLoc that are robust to outliers:

Robust FacLoc: In addition to F, C, opening costs $\{f_i | i \in F\}$, and metric d, we are also given an integer $0 \leq p \leq |C|$, that denotes the *coverage requirement.* The objective is to find a solution (C', F'), where $F' \subseteq F$, $C' \subseteq C$, with $|C'| \geq p$, and

$$\mathsf{cost}(C', F') := \sum_{i \in F'} f_i + \sum_{j \in C'} d(j, F')$$

is minimized over all (F', C'), where $|C'| \geq p$.

FacLoc with Penalties: In addition to F, C, opening costs $\{f_i | i \in F\}$, and metric d, we are also given penalties $p_j \geq 0$ for each client $j \in C$. The objective is to find a solution (C', F'), where $F' \subseteq F$, and $C' \subseteq C$, such that,

$$\mathsf{cost}(C', F') := \sum_{i \in F'} f_i + \sum_{j \in C'} d(j, F') + \sum_{j \in C \setminus C'} p_j$$

is minimized over all (C', F').

In this paper we present distributed $O(1)$-approximation algorithms for Robust FacLoc and FacLoc with Penalties in several models of large-scale distributed computation. As far as we know, these are the first distributed algorithms for versions of FacLoc that are robust to outliers. In distributed settings, the complexity of the problem can be quite sensitive to the manner in which input is specified. We consider two alternate ways of specifying the input to the problem.

Explicit metric: The metric d is specified *explicitly* as a $|F| \times |C|$ matrix distributed among the machines of the underlying communication network. This explicit description of the metric assumes that the $|F| \times |C|$ matrix fits in the total memory of all machines combined.

Implicit metric: In this version, the metric is specified *implicitly* – as the shortest path metric of a given edge-weighted graph whose vertex set is $C \cup F$; we call this the *metric graph*. The reason for considering this alternate specification of the metric is that it can be quite compact; the graph specifying the metric can be quite sparse (e.g., having $O(|F| + |C|)$ edges). Thus, in settings where $|F| \cdot |C|$ is excessively large, but $|F| + |C|$ is not, this is a viable option.

For the facility location problems considered in this paper, when the input metric is explicitly specified, the biggest challenge is solving the *maximal independent set (MIS)* problem efficiently. When the input metric is implicitly specified, the biggest challenge is to efficiently learn just enough of the metric space. Thus, changing the input specification changes the main challenge in a fundamental way and consequently we obtain very different results for the two alternate input specifications.

Our algorithms run in 3 models of distributed computation, which we now describe specifically in the context of facility location problems. All three models are synchronous message passing model.

Congested Clique model: The Congested Clique model was introduced by Lotker et al. [31] and then extensively studied in recent years [17, 18, 16, 9, 23, 26, 34, 13, 12, 37, 29]. In this model, the underlying communication network is a clique and the number of nodes in this clique equals $|F|+|C|$. In each round, each node performs local computation based on the information it holds and then sends a (possibly distinct) $O(\log n)$-size message to each of the remaining nodes. Initially, each node hosts a facility or a client and the node hosting facility i knows the opening cost f_i and the node hosting client j knows the penalty p_j for the FacLoc with penalties problem. In the explicit metric setting, the node hosting facility i knows all the connection costs $d(i, j)$ to all clients $j \in C$. Similarly, the node hosting client j knows all connection costs $d(i, j)$ to all facilities $i \in F$. In the implicit metric setting, the node hosting a facility or client knows the edges of the metric graph incident on that facility or client. We call this input distribution *vertex-centric* because each node is responsible for the local input of a facility or client. The vertex-centric assumption can be made without loss of generality because an adversarially (but evenly) distributed input can be redistributed in a vertex-centric manner among the nodes in constant rounds using Lenzen's routing protocol [29].

Massively Parallel Computation (MPC) model: The MPC model was introduced in [27] and variants of this model were considered in [19, 5, 2]. It can be viewed as a clean abstraction of the MapReduce model. We are given k machines, each with S words of space and the input is distributed in a vertex-centric fashion among the machines, the only difference being that machines can host multiple facilities and clients (provided they fit in memory). Let I be the total input size. Typically, we require k and S to each be sublinear in I, that is $O(I^{1-\varepsilon})$ for some $\varepsilon > 0$. We also require that the total memory not be too much larger than needed for the input, i.e., $k \times S = O(I)$. In each round, each machine sends and receives a total of $O(S)$ words of information because it is the volume of information that will fit into its memory. In our work we consider

MPC algorithms with memory $S = \tilde{O}(n)$ [1] where $n = |F| = |C|$. In the explicit metric setting, since $I = O(n^2)$, even if we assume $S = \tilde{O}(n)$, k and S are still strictly sublinear in I. But in the implicit metric setting, if we assume $S = \tilde{O}(n)$ then the memory may not be strictly sublinear in the input size when the input graph is sparse, having $O(n)$ edges for example. Therefore, our algorithms are not strictly MPC algorithms when the input is sparse. Similar to the Congested Clique model, we can assume that the input is distributed in a vertex-centric manner without loss of generality, due to the nature of communication in each round and the fact that $S = \Omega(n)$.

k-machine model: The k-machine model was introduced in [28] and further studied in [36]. This model abstracts essential features of systems such as Pregel [32] and Giraph (see `http://giraph.apache.org/`) that have been designed for large-scale graph processing. We are given k machines and the input is distributed among the machines. In [28], the k-machine model is used to solve graph problems and they assume a *random vertex partition* distribution of the input graph among the k machines. In other words, each vertex along with its incident edges is provided to one of the k machines chosen uniformly at random. The corresponding assumption for facility location problems would be that each facility and each client is assigned uniformly at random to one of the k machines. Facility $i \in F$ comes with its opening cost f_i and client $j \in C$ comes with its penalty p_j for the FacLoc with penalties problem. In the explicit metric setting, each facility $i \in F$ comes with connections costs $d(i, j)$ for all $j \in C$ whereas in the implicit metric setting facility i comes along with the edges of the metric graph incident on it. Similarly for each client $j \in C$. In each round, each machine can send a (possibly distinct) size-B message to each of the remaining $k - 1$ machines. Typically, B is assumed to be poly($\log n$) bits [28].

The Congested Clique model does not directly model settings of large-scale computation because in this model the number of nodes in the underlying communication network equals the number of vertices in the input graph. However, fast Congested Clique algorithms can usually be translated (sometimes automatically) to fast MPC and k-machine algorithms. So the Congested Clique algorithms in this paper are important stepping stones towards more complex MPC and k-machine algorithms [20, 28]. The MPC model and the k-machine model are quite similar. Even though the k-machine model is specified with a per-edge bandwidth constraint of B bits, it can be equivalently described with a per-machine bandwidth constraint of $k \cdot B$ bits that can be sent and received in each round. Thus setting $k \cdot B = S$ makes the k-machine model and MPC model equivalent in their bandwidth constraint. Despite their similarities, it is useful to think about both models due to differences in how they are parameterized and how these parameters affect the running times of algorithms in these models. For example, in the MPC model, usually one starts by picking S as a sublinear function of the input size n. This leads to the number of machines being fixed and the running time of the algorithm is expressed as a function of n. In the k-machine model B is usually fixed at poly($\log n$) and the running time of the algorithm is expressed as a function of n and k. This helps us understand how the running time changes as we increase k. For example, algorithms with running times of the form $O(n/k)$ exhibit a linear speedup as k increases, whereas algorithms with running time of the form $O(n/k^2)$ indicating a quadratic speedup [35].

[1] Throughout the paper, we use $\tilde{O}(f(n))$ as a shorthand for $O(f(n) \cdot \text{poly}(\log n))$ and $\tilde{\Omega}(f(n))$ as a shorthand for $\Omega(f(n)/\text{poly}(\log n))$.

1.1 Main Results

In order to obtain $O(1)$-approximation algorithms for Robust FacLoc and FacLoc with Penalties, Charikar et al. [10] propose modifications to the primal-dual approximation algorithm for FacLoc due to Jain and Vazirani [25]. The problem with using this approach for our purposes is that it seems difficult obtain fast *distributed* algorithms using the Jain-Vazirani approach. For example, obtaining a *sublogarithmic* round $O(1)$-approximation for FacLoc in the Congested Clique model using this approach seems difficult. However, as established in our previous work [21, 4, 8] and in [15] the greedy algorithm of Mettu and Plaxton [33] for FacLoc seems naturally suited for fast distributed implementation.

The first contribution of this paper is to show that $O(1)$-approximation algorithms to Robust FacLoc and FacLoc with Penalties can *also* be obtained by using variants of the Mettu-Plaxton greedy algorithm. Our second contribution is to show that by combining ideas from earlier work [21, 4] with some new ideas, we can efficiently implement distributed versions of the variants of the Mettu-Plaxton algorithm for Robust FacLoc and FacLoc with Penalties. The specific results we obtain for the two versions of input specification are as follows. For simplicity of exposition, we assume $|C| = |F| = n$.

- **Implicit metric**: For both problems, we present $O(1)$-approximation algorithms running in $O(\text{poly}(\log n))$ rounds in the Congested Clique and the MPC model. Assuming the metric graph has m edges, the input size is $\Theta(m + n)$ and we use $\tilde{O}(m/n)$ machines each with memory $\tilde{O}(n)$. In the k-machine model, we present $O(1)$-approximation algorithms running in $\tilde{O}(n/k)$ rounds.
- **Explicit metric**: For both problems, we present extremely fast $O(1)$-approximation algorithms, running in $O(\log \log \log n)$ rounds, in the Congested Clique and the MPC model. The input size is $\Theta(n^2)$ and we use n machines each with memory $\tilde{O}(n)$ in the MPC model. In the k-machine model, we present $O(1)$-approximation algorithms running in $\tilde{O}(n/k)$ rounds.

Due to space constraints we only describe our distributed implementations for the Robust FacLoc algorithm and omit most of the technical proofs. The full version with all the technical details appears in [24].

2 Sequential Algorithms for Facility Location with Outliers

We first describe the greedy sequential algorithm of Mettu and Plaxton [33] (Algorithm 1) for the Metric FacLoc problem which will serve as a building block for our algorithms for Robust FacLoc and the FacLoc with Penalties discussed in this section. The algorithm first computes a "radius" r_i for each facility $i \in F$ and it then greedily picks facilities to open in non-decreasing order of radii provided no previously opened facility is too close. The "radius" of a facility i is the amount that each client is charged for the opening of facility i. Clients pay towards this charge after paying towards the cost of connecting to facility i; clients that have a large connection cost to i pay nothing towards this charge. It is shown in [33] using a charging argument that Algorithm 1 is 3-approximation for the Metric FacLoc problem. Later on, [3] gave a primal-dual analysis, showing the same approximation guarantee, by comparing the cost of the solution to a dual feasible solution. We use the latter analysis approach as it can be easily modified to work for the algorithms with outliers.

Algorithm 1: FacilityLocationMP(F, C).

```
  /* Radius Computation Phase:                                              */
1 For each $i \in F$, compute $r_i \geq 0$, satisfying $f_i = \sum_{j \in C} \max\{0, r_i - c_{ij}\}$.
  /* Greedy Phase:                                                          */
2 Sort and renumber facilities in the non-decreasing order of $r_i$.
3 $F' \leftarrow \emptyset$                                          ▷ Solution set
4 for $i = 1, 2, \ldots$ do
5     if there is no facility in $F'$ within distance $2r_i$ from $i$ then
6         $F' \leftarrow F' \cup \{i\}$
7     end
8 end
9 Connect each client $j$ to its closest facility in $F'$.
```

2.1 Robust Facility Location

Since we use the primal dual analysis of [3] to get a bounded approximation factor, we need to address the fact that the standard linear programming relaxation for Robust FacLoc has unbounded integrality gap. To fix this we modify the instance in a similar manner to [10]. Let (C^*, F^*) be a fixed optimal solution, and let $i_* \in F$ be a facility in that solution with the maximum opening cost f_{i_*}. We begin by assuming that we are given a facility, say i_e with opening cost f_{i_e}, such that, $f_{i_*} \leq f_{i_e} \leq \alpha f_{i_*}$, where $\alpha \geq 1$ is a constant. Now, we modify the original instance by changing the opening costs of the facilities as follows.

$$f'_i = \begin{cases} +\infty & \text{if } f_i > f_{i_e} \\ 0 & \text{if } i = i_e \\ f_i & \text{otherwise} \end{cases}$$

Note that we can remove the facilities with opening cost $+\infty$ without affecting the cost of an optimal solution, and hence we assume that w.l.o.g. all the modified opening costs f'_i are finite.

Let (C^*_e, F^*_e) be an optimal solution for this modified instance, and let $\mathsf{cost}_e(C^*_e, F^*_e)$ be its cost using the modified opening costs. Observe that without loss of generality, we can assume that $i_e \in F^*_e$, since its opening cost f'_{i_e} equals 0. We obtain the following lemma and its simple corollary.

▶ **Lemma 1.** $\mathsf{cost}_e(C^*_e, F^*_e) \leq \mathsf{cost}(C^*, F^*)$.

▶ **Corollary 2.** *Let (C'_e, F'_e) be a feasible solution for the instance with modified facility opening costs, such that, $\mathsf{cost}_e(C'_e, F'_e) \leq \beta \cdot \mathsf{cost}_e(C^*_e, F^*_e) + \gamma \cdot f_{i_e}$ (where $\beta \geq 1, \gamma \geq 0$). Then, (C'_e, F'_e) is a $\beta + \alpha \cdot (\gamma + 1)$ approximation for the original instance.*

To efficiently find a facility i_e satisfying $f_{i_*} \leq f_{i_e} \leq \alpha f_{i_*}$, we partition the facilities into sets where each set contains facilities with opening costs from the range $\left[(1+\varepsilon)^i, (1+\varepsilon)^{i+1}\right)$. Iterating over all such ranges, and choosing a facility with highest opening cost from that range, we are guaranteed to find a facility i_e such that, $f_{i_*} \leq f_{i_e} \leq (1+\varepsilon) f_{i_*}$ [2]. The total

[2] An alternative approach would be to consider each facility one-by-one as a candidate, but for an efficient distributed implementation we can only afford $O(\log n)$ distinct guesses.

number of such iterations will be $O(\log_{1+\varepsilon} \frac{f_{\max}}{f_{\min}})$, where $f_{\max}$ is the largest opening cost, and $f_{\min}$ is the smallest non-zero opening cost. Assuming that every individual item in the input (e.g., facility opening costs, connection costs, etc.) can each be represented in $O(\log n)$ bits and that ε is a constant, this amounts to $O(\log n)$ iterations.

Our facility location algorithm is described in Algorithm 2. This algorithm can be thought of as running $O(\log n)$ separate instances of a modified version of the original Mettu-Plaxton algorithm (Algorithm 1), where in each instance of the Mettu-Plaxton algorithm, the algorithm is terminated as soon as the number of outlier clients drops below the required number, following which there is some post-processing.

We abuse the notation slightly, and denote by (C', F') the solution returned by the algorithm, i.e., the solution (C'_t, F'_t) corresponding to the iteration t of the outer loop that results in a minimum cost solution. Similarly, we denote by i_e a facility chosen in line 2 in the iteration corresponding to this iteration t.

▶ **Theorem 3.** $\mathsf{cost}_e(C', F') \leq 3 \cdot \mathsf{cost}_e(C^*_e, F^*_e) + f_{i_e}$

Applying Corollary 2 with $\alpha = 1 + \varepsilon, \beta = 3, \gamma = 1$ yields the following approximation guarantee.

▶ **Theorem 4.** *The solution returned by Algorithm 2 is a $5 + \varepsilon$ approximation to the Robust* FacLoc *problem.*

2.2 Facility Location with Penalties

For the penalty version, each client j comes with a penalty p_j which is the cost we pay if we make j an outlier. Therefore, the radius computation for a facility changes because if a facility i is asking client j to contribute more than $p_j - c_{ij}$ then it is cheaper for j to mark itself as an outlier and pay its penalty. Therefore, for each facility $i \in F$, let $r_i \geq 0$ be a value such that $f_i = \sum_{j \in C} \max\{\min\{r_i - c_{ij}, p_j - c_{ij}\}, 0\}$, if it exists. Notice that if for a facility $i \in F$, such an r_i does not exist, then it must be the case that for all $j \in C$, $p_j \leq c_{ij}$. That is, it is for any client, it is cheaper to pay the penalty than to connect it to this facility. Therefore, removing such a facility from consideration does not affect the cost of any solution, and hence we assume that for all $i \in F$, an $r_i \geq 0$ exists such that $f_i = \sum_{j \in C} \max\{\min\{r_i - c_{ij}, p_j - c_{ij}\}, 0\}$. The algorithm for FacLoc with Penalties is shown in Algorithm 3.

A primal-dual analysis of Algorithm 3 leads to the following upper bound.

▶ **Theorem 5.** $\mathsf{cost}(C', F') \leq 3 \cdot \mathsf{cost}(C^*, F^*)$.

3 Distributed Robust Facility Location: Implicit Metric

We first present our k-machine algorithm for Distributed Robust FacLoc in the implicit metric setting and derive the Congested Clique as a special case for $k = n$. We then describe how to implement the algorithm in the MPC model.

3.1 The k-Machine Algorithm

In this section we show how to implement the sequential algorithms for the Robust FacLoc in the k-machine model. To do this we first need to establish some primitives and techniques. These have largely appeared in [4]. Then we will provide details for implementing the Robust FacLoc algorithm in the k-machine model.

Algorithm 2: RobustFacLoc(F, C, p).

```
   /* Recall: $\ell := |C| - p$ */
1  for $t = 0, \ldots, O(\log n)$ do
2      Let $i_e \in F$ be the most expensive facility from the facilities with opening costs in
         the range $[(1+\varepsilon)^t, (1+\varepsilon)^{t+1})$ for some small constant $\varepsilon > 0$
3      Modify the facility opening costs to be
           $f'_i = \begin{cases} +\infty & \text{if } f_i > f_{i_e} \\ 0 & \text{if } i = i_e \\ f_i & \text{otherwise} \end{cases}$
       /* Radius Computation Phase: */
4      For each $i \in F$, compute $r_i \geq 0$, satisfying $f'_i = \sum_{j \in C} \max\{0, r_i - c_{ij}\}$.
       /* Greedy Phase: */
5      Sort and renumber facilities in the non-decreasing order of $r_i$.
6      Let $C' \leftarrow \emptyset$, $F' \leftarrow \emptyset$, $O' \leftarrow C$
7      Let $F_0 \leftarrow \emptyset$
8      for $i = 1, 2, \ldots$ do
9          if there is no facility in $F'$ within distance $2r_i$ from $i$ then
10             $F' \leftarrow F' \cup \{i\}$
11         end
12         $F_i \leftarrow F_{i-1} \cup \{i\}$
13         Let $C_i$ denote the set of clients that are within distance $r_i$
14         $C' \leftarrow C' \cup C_i$,    $O' \leftarrow O' \setminus C_i$.
15         if $|O'| \leq \ell$ then break
16     end
       /* Outlier Determination Phase: */
17     if $|O'| > \ell$ then
18         Let $O_1 \subseteq O'$ be a set of $|O'| - \ell$ clients that are closest to facilities in $F'$.
19         $C' \leftarrow C' \cup O_1$,    $O' \leftarrow O' \setminus O_1$.
20     end
21     else if $|O'| < \ell$ then
22         Let $O_2 \subseteq C'$ be the set of $\ell - |O'|$ clients with largest distance to open
             facilities $F'$.
23         $C' \leftarrow C' \setminus O_2$,    $O' \leftarrow O' \cup O_2$.
24     end
25     Let $(C'_t, F'_t) \leftarrow (C', F')$
26 end
27 return Return $(C'_t, F'_t)$ with the minimum cost.
```

Since the input metric is only implicitly provided, as an edge-weighted graph, a key primitive that we require is computing shortest path distances to learn parts of the metric space. To this end, the following lemma shows that we can solve the Single Source Shortest Paths (SSSP) problem efficiently in the k-machine model.

▶ **Lemma 6** (Corollary 1 in [4]). *For any $0 < \varepsilon \leq 1$, there is a deterministic $(1+\varepsilon)$-approximation algorithm in the k-machine model for solving the SSSP problem in undirected graphs with non-negative edge-weights in $O((n/k) \cdot poly(\log n)/poly(\varepsilon))$ rounds.*

Algorithm 3: PenaltyFacLoc(F, C, p).

```
   /* Radius Computation Phase: */
1  Compute $r_i$ for each $i \in F$ satisfying $f_i = \sum_{j \in C} \max\{\min\{r_i - c_{ij}, p_j - c_{ij}\}, 0\}$.
   /* Greedy Phase: */
2  Sort and renumber facilities in the non-decreasing order of $r_i$.
3  $C' \leftarrow \emptyset$,    $F' \leftarrow \emptyset$,    $O' \leftarrow \emptyset$.
4  for $i = 1, 2, \ldots$ do
5      if there is no facility in $F'$ within distance $2r_i$ from $i$ then
6          $F' \leftarrow F' \cup \{i\}$
7      end
8  end
   /* Outlier Determination Phase: */
9  for each client $j$ do
10     Let $i$ be the closest facility to $j$ in $F'$
11     if $c_{ij} \le p_j$ then $C' \leftarrow C' \cup \{j\}$
12     else $O' \leftarrow O' \cup \{j\}$
13 end
14 return $(C', F')$ as the solution.
```

Algorithm 4: RadiusComputation Algorithm.

1 **Neighborhood-Size Computation.** Each machine m_j computes $q_i(v)$, for all integers $i \ge 0$ and for all vertices $v \in H(m_j)$.

2 **Local Computation.** Each machine m_j computes $\tilde{r}_v$ locally, for all vertices $v \in H(m_j)$. (Recall that $\tilde{r}_v := (1+\varepsilon)^{t-1}$ where $t \ge 1$ is the smallest integer for which $\sum_{i=0}^{t} q_i(v) \cdot ((1+\varepsilon)^{i+1} - (1+\varepsilon)^i) > f_v$.)

In addition to SSSP, our algorithms require an efficient solution to a more general problem that we call *Multi-Source Shortest Paths* (in short, MSSP). The input is an edge-weighted graph $G = (V, E)$, with non-negative edge-weights, and a set $T \subseteq V$ of sources.

For MSSP, the output is required to be, for each vertex v, the distance $d(v, T)$ (i.e., $\min\{d(v, u) \mid u \in T\}$) and the vertex $v^* \in T$ that realizes this distance. The following lemma shows that we can solve this problem efficiently in the k-machine model.

▶ **Lemma 7** (Lemma 4 in [4]). *Given a set $T \subseteq V$ of sources known to the machines (i.e., each machine m_j knows $T \cap H(m_j)$), we can, for any value $0 \le \varepsilon \le 1$, compute a $(1+\varepsilon)$-approximation to MSSP in $\tilde{O}(1/poly(\varepsilon) \cdot n/k)$ rounds, w.h.p. Specifically, after the algorithm has ended, for each $v \in V \setminus T$, the machine m_j that hosts v knows a pair $(u, \tilde{d}) \in T \times \mathbb{R}^+$, such that $d(v, u) \le \tilde{d} \le (1+\varepsilon) \cdot d(v, T)$.*

Using the primitives described above, [4] show that it is possible to compute approximate radius values efficiently in the k-machine model. The algorithm is described here, see the full version [24] for details on the implementation of this algorithm.

For any facility or client v and for any integer $i \ge 1$, let $q_i(v)$ denote $|B(v, (1+\varepsilon)^i)|$, the size of the neighborhood of v within distance $(1+\varepsilon)^i$.

Therefore, we get the following lemma the proof of which can be found in Section 4 of [4].

▶ **Lemma 8.** *For each facility $v \in F$ it is possible to compute an approximate radius $\tilde{r}_v$ in $\tilde{O}(n/k)$ rounds of the k-machine model such that $\frac{r_v}{(1+\varepsilon)^2} \leq \tilde{r}_v \leq (1+\varepsilon)^2 r_v$ where r_v is the actual radius of v satisfying $f_v = \sum_{u \in B(v,r_v)}(r_v - d(v,u))$.*

The greedy phase is implemented by discretizing the radius values computed in the first phase which results in $O(\log_{1+\varepsilon} n)$ distinct categories. Note that in each category, the order in which we process the facilities does not matter as it will only add an extra $(1+\varepsilon)$ factor to the approximation ratio. This reduces the greedy phase to computing a *maximal independent set (MIS)* on a suitable intersection graph for each category i where the vertices are the facilities in the i^{th} category and there is an edge between two vertices if they are within distance $2(1+\varepsilon)^i$ of each other.

Finding such an MIS requires $O(\log n)$ calls to a subroutine that solves MSSP [39] and since our implementation of MSSP only returns approximate distances, what we really compute is a relaxed version of an MIS called an (ε, d)-MIS in [4].

▶ **Definition 9** ((ε, d)-approximate MIS)**.** For an edge-weighted graph $G = (V, E)$, and parameters $d, \varepsilon > 0$, an (ε, d)-approximate MIS is a subset $I \subseteq V$ such that

1. For all distinct vertices $u, v \in I$, $d(u,v) \geq \frac{d}{1+\varepsilon}$.
2. For any $u \in V \setminus I$, there exists a $v \in I$ such that $d(u,v) \leq d \cdot (1+\varepsilon)$.

The work in [4] gives an algorithm that efficiently computes an approximate MIS of an induced subgraph $G[W]$ of G for any vertex set W in the k-machine model.

▶ **Lemma 10.** *We can find an $(O(\varepsilon), d)$-approximate MIS I of $G[W]$ whp in $\tilde{O}(n/k)$ rounds.*

We are now ready to describe the k-machine model implementation of Algorithm 2.

Our k-machine model implementation of the Robust FacLoc algorithm is summarized in Algorithm 5. The correctness proof is similar to that of Algorithm 2 but is complicated by the fact that we compute $(1+\varepsilon)$-approximate distances instead of exact distances. Again, as in the analysis of the sequential algorithm, we abuse the notation so that (i) (C', F') refers to a minimum-cost solution returned by the algorithm, (ii) i_e refers to the facility chosen in the line 2 of the algorithm, and (iii) the modified instance with original facility costs. This analysis appears in the full version [24], and as a result we get the following theorem.

▶ **Theorem 11.** *In $\tilde{O}(poly(1/\varepsilon) \cdot n/k)$ rounds, whp, Algorithm 5 finds a factor $5 + O(\varepsilon)$ approximate solution (C', F') to the Robust* FacLoc *problem for any constant $\varepsilon > 0$.*

3.2 The Congested Clique and MPC Algorithms

The algorithm for Congested Clique is essentially the same as the k-machine model algorithm with $k = n$. The only technical difference is that in the k-machine model, the input graph vertices are randomly partitioned across the machines. This means that even though there are n vertices and n machines, a single machine may be hosting multiple vertices. It is easy to see that the Congested Clique model, in which each machine holds exactly one vertex can simulate the k-machine algorithm with no overhead in rounds. Therefore, by substituting $k = n$ in the running time of Theorem 11, we get the following result.

▶ **Theorem 12.** *In $O(\text{poly}\log n)$ rounds of Congested Clique, whp, we can find a factor $5 + O(\varepsilon)$ approximate solution to the Robust* FacLoc *problem for any constant $\varepsilon > 0$.*

Now we focus on the implementing the MPC algorithm. The first crucial observation is that Algorithm 5 reduces the task of finding an approximate solution to the Robust FacLoc

Algorithm 5: RobustFacLocDist(F, C, p).

```
   /* Recall $\ell := |C| - p$                                                */
1  for $t = 1, \dots, O(\log n)$ do
2      Let $i_e \in F$ be a most expensive facility from the facilities with opening costs in
        the range $\left[(1+\varepsilon)^t, (1+\varepsilon)^{t+1}\right)$
3      Modify the facility opening costs to be
           $$f'_i = \begin{cases} +\infty & \text{if } f_i > f_{i_e} \\ 0 & \text{if } i = i_e \\ f_i & \text{otherwise} \end{cases}$$
       /* Radius Computation Phase:                                           */
4      Call the RadiusComputation algorithm (Algorithm 4) to compute approximate
        radii.
       /* Greedy Phase:                                                       */
5      Let $F' = \emptyset$, $C' = \emptyset$, $O' = C$
6      for $i = 0, 1, 2, \dots$ do
7          Let $W$ be the set of vertices $w \in F$ across all machines with $\tilde{r}_w = \tilde{r} = (1+\varepsilon)^i$
8          Using Lemma 7, remove all vertices from $W$ within approximate distance
            $2(1+\varepsilon)^3 \cdot \tilde{r}$ from $F'$
9          $I \leftarrow$ ApproximateMIS$(G, W, 2(1+\varepsilon)^3 \cdot \tilde{r}, \varepsilon)$
10         $F' \leftarrow F' \cup I$
11         Using Lemma 7, move from $O'$ to $C'$ all vertices that are within distance
            $(1+\varepsilon) \cdot \tilde{r}$ from $F_i$, the set of facilities processed up to iteration $i$
12         if $|O'| \leq \ell$ then break
13     end
       /* Outlier Determination Phase:                                        */
14     if $|O'| > \ell$ then
15         Using Lemma 7 find $O_1 \subseteq O'$, a set of $|O'| - \ell$ clients that are closest to
            facilities in $F'$.
16         $C' \leftarrow C' \cup O_1$,    $O' \leftarrow O' \setminus O_1$.
17     end
18     else if $|O'| < \ell$ then
19         Using Lemma 7 find $O_2 \subseteq C \setminus O'$, a set of $(\ell - |O'|)$ clients that are farthest
            away from facilities in $F'$
20         $C' \leftarrow C' \setminus O_2$,    $O' \leftarrow O' \cup O_2$.
21     end
22     Let $(C'_t, F'_t) \leftarrow (C', F')$
23 end
24 return $(C'_t, F'_t)$ with a minimum cost
```

problem in the implicit metric setting to poly $\log n$ calls to a $(1+\varepsilon)$-approximate SSSP subroutine along with some local bookkeeping. Therefore, all we need to do is efficiently implement an approximate SSSP algorithm in the MPC model.

The second fact that helps us is that Becker et al. [6] provide a distributed implementation of their approximate SSSP algorithm in the Broadcast Congested Clique (BCC) model. The

BCC model is the same as the Congested Clique model but with the added restriction that nodes can only broadcast messages in each round. Therefore we get the following simulation theorem, which follows almost immediately from Theorem 3.1 of [7].

▶ **Theorem 13.** *Let $\mathcal{A}$ be a T round BCC algorithm that uses $\tilde{O}(n)$ local memory at each node. One can simulate $\mathcal{A}$ in the MPC model in $O(T)$ rounds using $\tilde{O}(n)$ memory per machine.*

In any T round BCC algorithm, each vertex will receive $O(n \cdot T)$ distinct messages. The approximate SSSP algorithm of Becker et al. [6] runs in $O(\text{poly}\log n/\,\text{poly}(\varepsilon))$ rounds and therefore, uses $\tilde{O}(n)$ memory per node to store all the received messages (and for local computation). Therefore, we get the following theorem.

▶ **Theorem 14.** *In $O(\text{poly}\log n)$ rounds of MPC, whp, we can find a factor $5 + O(\varepsilon)$ approximate solution to the Robust* FacLoc *problem for any constant $\varepsilon > 0$.*

4 Distributed Robust Facility Location: Explicit Metric

For the k-machine model implementation, the implicit metric algorithm from the previous section also provides a similar guarantee for the explicit metric setting and hence we do not discuss it separately in this section.

4.1 The Congested Clique Algorithm

The work in [21] presents a Congested Clique algorithm that runs in expected $O(\log\log n)$ rounds and computes an $O(1)$-approximation to FacLoc. This is improved exponentially in [22] which presents an $O(1)$-approximation algorithm to FacLoc running in $O(\log\log\log n)$ rounds whp. The algorithms in [21] and in [22] are essentially the same with one key difference. They both reduce the problem of solving FacLoc in the Congested Clique model to the ruling set problem. Specifically, showing that if a t-ruling set can be computed in T rounds, then an $O(t)$-approximation to FacLoc can be computed in $O(T)$ rounds. In [21] a 2-ruling set is computed in expected $O(\log\log n)$ rounds, whereas in [22] it is computed in $O(\log\log\log n)$ rounds whp.

The algorithm for computing an $O(1)$-approximation to Robust FacLoc (see Section 2.1) is essentially the FacLoc algorithm in [21, 22], but with an outer loop that runs $O(\log n)$ times. In each iteration of this outer loop, we modify the facility opening costs in a certain way and solve FacLoc on the resulting instance. Thus we have $O(\log n)$ instances of FacLoc to solve and via the reduction in [21, 22], we have $O(\log n)$ independent instances of the ruling set problem to solve. Here we show that $O(\log n)$ independent instances of the $O(\log\log\log n)$-round 2-ruling set algorithm in [22] can be executed in parallel in the Congested Clique model, still in $O(\log\log\log n)$ rounds whp. To be precise, suppose that the input consists of $c = O(\log n)$ graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_c = (V, E_c)$.

▶ **Theorem 15.** *2-ruling sets for all graphs G_i, $1 \le i \le c$, can be computed in $O(\log\log\log n)$ rounds whp.*

The theorem above and the discussion preceding it leads to the following theorem.

▶ **Theorem 16.** *There is an $O(1)$-approximation algorithm in the Congested Clique model for Robust* FacLoc*, running in $O(\log\log\log n)$ rounds whp.*

4.2 The MPC Algorithm

We now utilize the Congested Clique algorithm for Robust FacLoc to design an MPC model algorithm for Robust FacLoc, also running in $O(\log\log\log n)$ rounds whp. Since each vertex has explicit knowledge of n distances, the overall memory is $O(n^2)$ words. Since the memory of each machine is $\tilde{O}(n)$, the number of machines will be $\tilde{O}(n)$ as well. Therefore, we can simulate the algorithm from the preceding section using Theorem 3.1 of [7] in the MPC model. We summarize our result in the following theorem.

▶ **Theorem 17.** *There is an $O(1)$-approximation algorithm for Robust* FacLoc *that can be implemented in the MPC model with $\tilde{O}(n)$ words per machine in $O(\log\log\log n)$ rounds whp.*

5 Conclusion and Open Questions

This paper presents fast $O(1)$-factor distributed algorithms for Facility Location problems that are robust to outliers. These algorithms run in the Congested Clique model and two models of large-scale computation, namely, the MPC model and the k-machine model. As far as we know these are the the first such algorithms for these important clustering problems.

Fundamental questions regarding the optimality of our results remain open. In the explicit metric setting, we present algorithms in the Congested Clique model and the MPC model that run in $O(\log\log\log n)$ rounds. While these may seem extremely fast, it is not clear that they are optimal. Via the results of Drucker et al. [13], it seems like showing a non-trivial lower bound in the Congested Clique model is out of the question for now. So a tangible question one can ask is whether we can further improve the running time of the 2-ruling set algorithm in the Congested Clique model, possibly solving it in $O(\log^* n)$ or even $O(1)$ rounds. This would immediately imply a corresponding improvement in the running time of our Congested Clique and MPC model algorithms in the explicit metric setting.

All the k-machine algorithms we present in the paper run in $\tilde{O}(n/k)$ rounds. It is unclear if this is optimal. In previous work [4], we showed a lower bound of $\tilde{\Omega}(n/k)$ in the implicit metric setting, assuming that in the output to facility location problems every open facility needed to know all clients that connect to it. The lower bound heavily relies on the implicit metric and the output requirement assumptions. However, even if we relax both of these assumptions, i.e., we work in the explicit metric setting and only ask that every client know the facility that will serve it, we still seem to be unable to get over the $\tilde{O}(n/k)$ barrier.

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 20–29, New York, NY, USA, 1996. ACM. doi:10.1145/237814.237823.

2 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel Algorithms for Geometric Graph Problems. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 574–583, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591805.

3 Aaron Archer, Ranjithkumar Rajagopalan, and David B. Shmoys. Lagrangian Relaxation for the k-Median Problem: New Insights and Continuity Properties. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings*, pages 31–42, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. doi:10.1007/978-3-540-39658-1_6.

4 Sayan Bandyapadhyay, Tanmay Inamdar, Shreyas Pai, and Sriram V. Pemmaraju. Near-Optimal Clustering in the k-machine model. In *Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN 2018, Varanasi, India, January 4-7, 2018*, pages 15:1–15:10, 2018. doi:10.1145/3154273.3154317.

5 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication Steps for Parallel Query Processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '13, pages 273–284, New York, NY, USA, 2013. ACM. doi:10.1145/2463664.2465224.

6 Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing (DISC 2017)*, volume 91 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.DISC.2017.7.

7 Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Brief Announcement: Semi-MapReduce Meets Congested Clique. *CoRR*, abs/1802.10297, 2018. arXiv:1802.10297.

8 Andrew Berns, James Hegeman, and Sriram V. Pemmaraju. Super-Fast Distributed Algorithms for Metric Facility Location. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 428–439, 2012. doi:10.1007/978-3-642-31585-5_39.

9 Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic Methods in the Congested Clique. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, pages 143–152, New York, NY, USA, 2015. ACM. doi:10.1145/2767386.2767414.

10 Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for Facility Location Problems with Outliers. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 642–651, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

11 Jeffrey Dean and Sanjay Ghemawat. MapReduce: A Flexible Data Processing Tool. *Commun. ACM*, 53(1):72–77, January 2010. doi:10.1145/1629175.1629198.

12 Danny Dolev, Christoph Lenzen, and Shir Peled. "Tri, Tri Again": Finding Triangles and Small Subgraphs in a Distributed Setting. In *Proceedings of the 26th International Symposium on Distributed Computing (DISC)*, pages 195–209, 2012.

13 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. The communication complexity of distributed task allocation. In *Proceedings of the 30st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 67–76, 2012.

14 Alina Ene, Sungjin Im, and Benjamin Moseley. Fast Clustering Using MapReduce. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 681–689, New York, NY, USA, 2011. ACM. doi:10.1145/2020408.2020515.

15 Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Mauro Sozio. Scalable Facility Location for Massive Graphs on Pregel-like Systems. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 273–282, New York, NY, USA, 2015. ACM. doi:10.1145/2806416.2806508.

16 Joachim Gehweiler, Christiane Lammersen, and Christian Sohler. A Distributed $O(1)$-approximation Algorithm for the Uniform Facility Location Problem. In *Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 237–243, 2006. doi:10.1145/1148109.1148152.

17 Mohsen Ghaffari. Distributed MIS via All-to-All Communication. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 141–149, 2017. `doi:10.1145/3087801.3087830`.

18 Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 129–138, 2018. `doi:10.1145/3212734.3212743`.

19 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, Searching, and Simulation in the Mapreduce Framework. In *Proceedings of the 22nd International Conference on Algorithms and Computation*, ISAAC'11, pages 374–383, Berlin, Heidelberg, 2011. Springer-Verlag. `doi:10.1007/978-3-642-25591-5_39`.

20 James W. Hegeman and Sriram V. Pemmaraju. Lessons from the Congested Clique applied to MapReduce. *Theor. Comput. Sci.*, 608:268–281, 2015. `doi:10.1016/j.tcs.2015.09.029`.

21 James W. Hegeman and Sriram V. Pemmaraju. Sub-logarithmic distributed algorithms for metric facility location. *Distributed Computing*, 28(5):351–374, 2015. `doi:10.1007/s00446-015-0243-x`.

22 James W. Hegeman, Sriram V. Pemmaraju, and Vivek Sardeshmukh. Near-Constant-Time Distributed Algorithms on a Congested Clique. *CoRR*, abs/1408.2071, 2014. `arXiv:1408.2071`.

23 Stephan Holzer and Nathan Pinsker. Approximation of Distances and Shortest Paths in the Broadcast Congest Clique. *arXiv preprint, arXiv:1412.3445*, 2014.

24 Tanmay Inamdar, Shreyas Pai, and Sriram V. Pemmaraju. Large-Scale Distributed Algorithms for Facility Location with Outliers. *CoRR*, abs/1811.06494, November 2018. `arXiv:1811.06494`.

25 Kamal Jain and Vijay V. Vazirani. Approximation Algorithms for Metric Facility Location and k-Median Problems Using the Primal-dual Schema and Lagrangian Relaxation. *J. ACM*, 48(2):274–296, March 2001. `doi:10.1145/375827.375845`.

26 Tomasz Jurdziński and Krzysztof Nowicki. MST in O(1) Rounds of Congested Clique. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 2620–2632, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics.

27 Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A Model of Computation for MapReduce. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 938–948, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

28 Hartmut Klauck, Danupon Nanongkai, Gopal Pandurangan, and Peter Robinson. Distributed Computation of Large-scale Graph Problems. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 391–410, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

29 Christoph Lenzen. Optimal Deterministic Routing and Sorting on the Congested Clique. In *Proceedings of the 31st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 42–50, 2013. `doi:10.1145/2484239.2501983`.

30 Shi Li. A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem. In *Proceedings of the 38th International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'11, pages 77–88, Berlin, Heidelberg, 2011. Springer-Verlag.

31 Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-Weight Spanning Tree Construction in $O(\log\log n)$ Communication Rounds. *SIAM Journal on Computing*, 35(1):120–131, 2005.

32 Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A System for Large-scale Graph Processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM. doi:10.1145/1807167.1807184.

33 Ramgopal R. Mettu and C. Greg Plaxton. The Online Median Problem. *SIAM J. Comput.*, 32(3):816–832, March 2003. doi:10.1137/S0097539701383443.

34 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proceedings of the 46th ACM Symposium on Theory of Computing (STOC)*, pages 565–573, 2014.

35 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. Fast Distributed Algorithms for Connectivity and MST in Large Graphs. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '16, pages 429–438, New York, NY, USA, 2016. ACM. doi:10.1145/2935764.2935785.

36 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. On the Distributed Complexity of Large-Scale Graph Computations. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 405–414, 2018. doi:10.1145/3210377.3210409.

37 Boaz Patt-Shamir and Marat Teplitsky. The Round Complexity of Distributed Sorting. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 249–256, 2011. doi:10.1145/1993806.1993851.

38 Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. Data Stream Clustering: A Survey. *ACM Comput. Surv.*, 46(1):13:1–13:31, July 2013. doi:10.1145/2522968.2522981.

39 Mikkel Thorup. Quick k-Median, k-Center, and Facility Location for Sparse Graphs. *SIAM Journal on Computing*, 34(2):405–432, 2005. doi:10.1137/S0097539701388884.